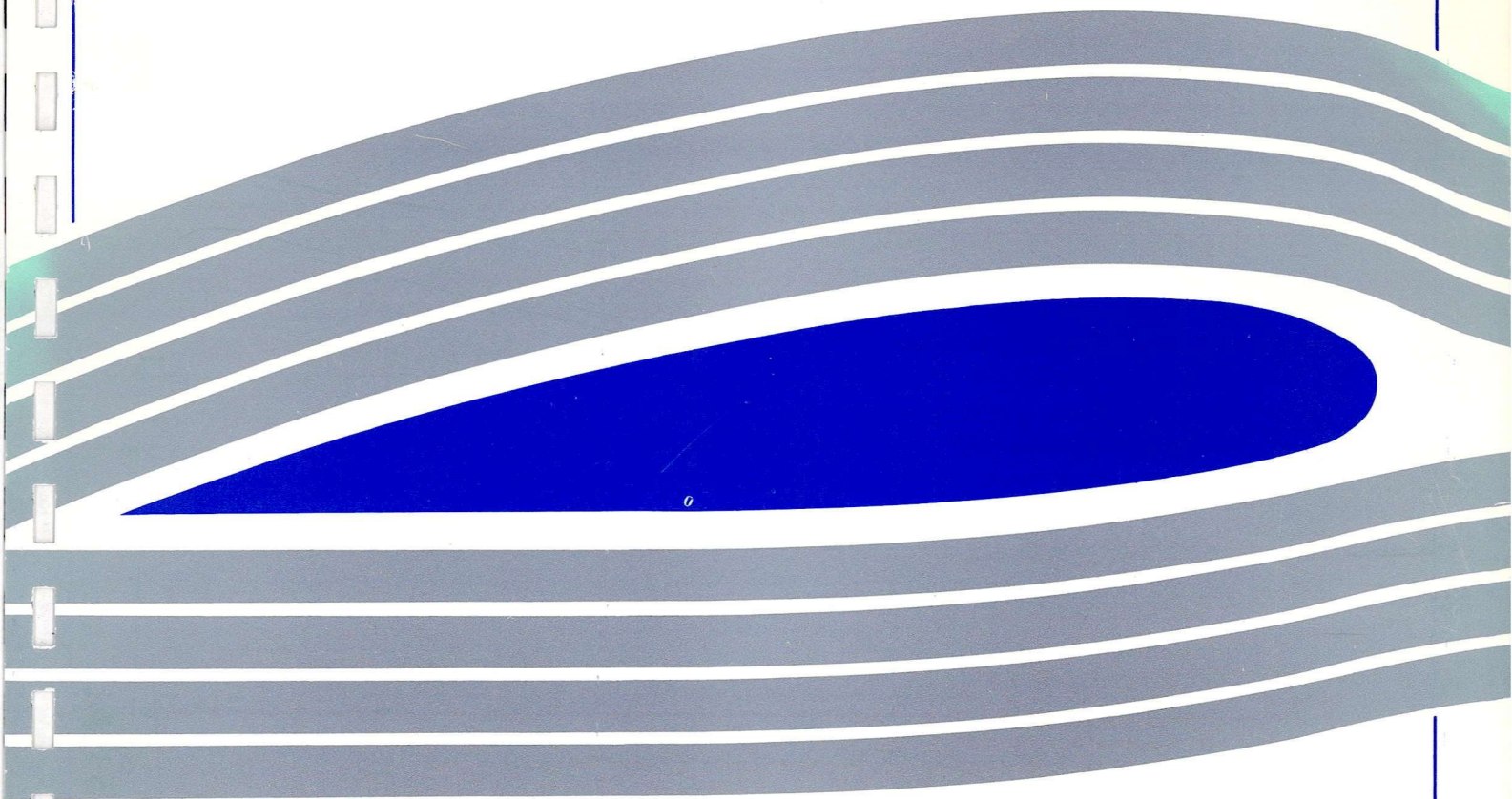
University of Glasgow
DEPARTMENT OF
AEROSPACE
ENGINEERING

A Three Dimensional Navier-Stokes Equations Solver with Turbulence Modelling

Xiaoshi Jin

GU Aero Report 9319, August, 1993



A Three Dimensional Navier-Stokes Equations Solver with Turbulence Modelling

Xiaoshi Jin

GU Aero Report 9319, August, 1993

CONTENTS

A Three Dimensional Navier-Stokes Equations Solver with Turbulence Modelling

Xiaoshi Jin

Department of Aerospace Engineering
University of Glasgow

August, 1993

1.INTRODUCTION

A three dimensional Navier-Stokes equations solver with turbulence modelling — NS3DT has been developed successfully based on Qin's original NS3D code[1]. The Osher's upwind scheme is used in this explicit finite volume algorithm which has the capacity to tackle both laminar and turbulent external supersonic and/or hypersonic viscous flows around a spacecraft shape.

This report gives a brief description of the method used in the solver, the validation of the code and the listing of the code itself. A user guide is also provided so that either further development or practical application of the code can be easily done.

2.FORMULATIONS

2.1 Governing Equations

The nondimensional strong conservation form of the full set of Navier-Stokes equations in Cartesian coordinates are listed as follows:

$$\frac{\partial Q}{\partial t} + \frac{\partial (E_i - E_v)}{\partial x} + \frac{\partial (F_i - F_v)}{\partial y} + \frac{\partial (G_i - G_v)}{\partial z} = 0 \qquad (2.1.1)$$

where the column hypervectors

$$Q = (\rho, \rho u, \rho v, \rho w, E_t)^T \qquad (2.1.2a)$$

$$E_i = \left(\rho u, \rho u^2 + p, \rho uv, \rho uw, (E_t + p)u\right)^T \qquad (2.1.2b)$$

$$F_i = \left(\rho v, \rho vu, \rho v^2 + p, \rho vw, (E_t + p)v\right)^T \qquad (2.1.2c)$$

$$G_i = \left(\rho w, \rho wu, \rho wv, \rho w^2 + p, (E_t + p)w\right)^T \qquad (2.1.2d)$$

$$E_v = \left(0, \tau_{xx}, \tau_{xy}, \tau_{xz}, u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + q_x\right)^T \qquad (2.1.2e)$$

$$F_v = \left(0, \tau_{yx}, \tau_{yy}, \tau_{yz}, u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + q_y\right)^T \qquad (2.1.2f)$$

$$G_v = \left(0, \tau_{zx}, \tau_{yz}, \tau_{zz}, u\tau_{zx} + v\tau_{yz} + w\tau_{zz} + q_z\right)^T \qquad (2.1.2g)$$

in which

$$E_t = \frac{p}{\gamma - 1} + \rho\frac{u^2 + v^2 + w^2}{2} \tag{2.1.3a}$$

$$\tau_{xx} = \frac{2(\mu + \mu_t)}{3\,Re}\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} - \frac{\partial w}{\partial z}\right) \tag{2.1.3b}$$

$$\tau_{yy} = \frac{2(\mu + \mu_t)}{3\,Re}\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x} - \frac{\partial w}{\partial z}\right) \tag{2.1.3c}$$

$$\tau_{zz} = \frac{2(\mu + \mu_t)}{3\,Re}\left(2\frac{\partial w}{\partial z} - \frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) \tag{2.1.3d}$$

$$\tau_{xy} = \tau_{yx} = \frac{(\mu + \mu_t)}{Re}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{2.1.3e}$$

$$\tau_{xz} = \tau_{zx} = \frac{(\mu + \mu_t)}{Re}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \tag{2.1.3f}$$

$$\tau_{yz} = \tau_{zy} = \frac{(\mu + \mu_t)}{Re}\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \tag{2.1.3g}$$

$$q_x = \frac{1}{(\gamma - 1)M_\infty^2\,Re}\left(\frac{\mu}{P_r} + \frac{\mu_t}{P_{rt}}\right)\frac{\partial T}{\partial x} \tag{2.1.3h}$$

$$q_y = \frac{1}{(\gamma - 1)M_\infty^2\,Re}\left(\frac{\mu}{P_r} + \frac{\mu_t}{P_{rt}}\right)\frac{\partial T}{\partial y} \tag{2.1.3i}$$

$$q_z = \frac{1}{(\gamma - 1)M_\infty^2\,Re}\left(\frac{\mu}{P_r} + \frac{\mu_t}{P_{rt}}\right)\frac{\partial T}{\partial z} \tag{2.1.3j}$$

where ρ, p, u, v, w, T are the nondimensional density, pressure, three components of the velocity in the x, y, z directions and the temperature, respectively; E_t is the total energy, τ the viscous stress tensor and six components of which are listed, μ the molecular coefficient of viscosity, μ_t the turbulent coefficient of viscosity, P_r the Prandtl number and P_{rt} the turbulent Prandtl number, γ is the ratio of specific heats, and M_∞ the free stream Mach number. The perfect gas equation of state is used which has the following form between temperature, pressure and density.

$$T = \gamma M_\infty^2 \frac{p}{\rho} \tag{2.1.4}$$

The molecular coefficient of viscosity is calculated using Sutherland equation:

$$\mu(T) = \begin{cases} T^{3/2}\left(\dfrac{\tilde{T}_\infty + 110.4}{T\tilde{T}_\infty + 110.4}\right) & T > D \\ \mu(D)\dfrac{T}{D} & T \le D \end{cases} \tag{2.1.5}$$

where $\tilde{T}_\infty$ is the free stream temperature.

In order to give the numerical equations which are based on the grid coordinate system, a space transformation from the Cartesian coordinate system to a local coordinate system must be introduced:

$$\left.\begin{aligned}\xi &= \xi(x,y,z)\\ \eta &= \eta(x,y,z)\\ \zeta &= \zeta(x,y,z)\end{aligned}\right\} \tag{2.1.6}$$

The governing equations can then be written in the following conservation law form:

$$\frac{\partial \tilde{Q}}{\partial t}+\frac{\partial(\tilde{E}_i-\hat{E}_v)}{\partial \xi}+\frac{\partial(\tilde{F}_i-\hat{F}_v)}{\partial \eta}+\frac{\partial(\tilde{G}_i-\hat{G}_v)}{\partial \zeta}=0 \tag{2.1.7}$$

where

$$\left.\begin{aligned}\tilde{Q} &= \frac{Q}{J}\\ \tilde{E}_i-\hat{E}_v &= \frac{\xi_x}{J}(E_i-E_v)+\frac{\xi_y}{J}(F_i-F_v)+\frac{\xi_z}{J}(G_i-G_v)\\ \tilde{F}_i-\hat{F}_v &= \frac{\eta_x}{J}(E_i-E_v)+\frac{\eta_y}{J}(F_i-F_v)+\frac{\eta_z}{J}(G_i-G_v)\\ \tilde{G}_i-\hat{G}_v &= \frac{\zeta_x}{J}(E_i-E_v)+\frac{\zeta_y}{J}(F_i-F_v)+\frac{\zeta_z}{J}(G_i-G_v)\end{aligned}\right\} \tag{2.1.8}$$

and J is the Jacobian matrix of the transformation given by

$$J=\frac{\partial(\xi,\eta,\zeta)}{\partial(x,y,z)} \tag{2.1.9}$$

2.2 Reynolds-Averaged Navier-Stokes Equations

By averaging the conservation laws over a time interval T, Reynolds in 1883 proposed the averaged Navier-Stokes equations for turbulent flows. This time interval is chosen large enough with respect to the time scale of turbulent fluctuations, but has to remain small with respect to the time scales of other time-dependent effects. The physical quantities are therefore defined by two parts:

$$A=\overline{A}+A' \tag{2.2.1}$$

where

$$\overline{A}(\vec{x},t)=\frac{1}{T}\int_{-T/2}^{T/2}A(\vec{x},t+\tau)\,d\tau \tag{2.2.2}$$

is the mean turbulent-averaged value, and A' is the fluctuation. For compressible flows, the density-weighted average is also used, which is defined through

$$\tilde{A} = \frac{\overline{\rho A}}{\bar{\rho}} \tag{2.2.3}$$

with

$$A = \tilde{A} + A'' \tag{2.2.4}$$

The continuity equation remains the same form for these turbulent averaged quantities while the momentum equations lead to the introduction of the Reynolds stress tensor, to be added to the averaged viscous stresses, and the energy equation brings up a turbulent heat flux vector into the diffusive flux term. Both the Reynolds stress tensor and the turbulent heat flux vector are given as follows:

$$\tau_{ij}^{R} = -\overline{\rho v_i'' v_j''} \tag{2.2.5}$$

$$\vec{F}_D^{T} = -\frac{\mu c_p}{P_r}\overline{\vec{\nabla} T} + \overline{\rho h'' \vec{v}''} \tag{2.2.6}$$

Except for these two extra terms, the Reynolds averaged equations have no difference with those for the laminar flows. However, Reynolds did not give a set of complete equations for both averaged and fluctuated physical quantities. For the solution to the turbulent flows, the uncompleted equations require additional information about the unknowns, which leads to the introduction of modellization of these two extra terms.

2.3 Baldwin-Lomax Turbulence Model

The of Baldwin-Lomax[2] turbulence model is used in the solver for the turbulent flow simulation. This model is a variant of the model proposed by Cebeci and Smith[3] in which the additional "effective" eddy viscosity for the effects of turbulence is calculated by an algebraic method. The model is a two-layer type, with μ_t given by

$$\mu_t = \begin{cases} (\mu_t)_{inner} & y \le y_{crossover} \\ (\mu_t)_{outer} & y > y_{crossover} \end{cases} \tag{2.3.1}$$

where y is the normal distance from the wall, and $y_{crossover}$ represents the nearest y at which the viscosities calculated by both inner and outer layer formulas are equal.
The viscosity in the inner layer is calculated by the Prandtl-Van Driest formulation

$$(\mu_t)_{inner} = \rho \, Re \, L^2 |\omega| \tag{2.3.2}$$

where

$$L = ky\left[1 - \exp\left(-y^+ / A^+\right)\right] \tag{2.3.3}$$

and $|\omega|$ is the magnitude of the vorticity, which is given by

$$|\omega| = \sqrt{\left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial z} - \frac{\partial w}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial x} - \frac{\partial u}{\partial z}\right)^2} \tag{2.3.4}$$

The friction height, y^+, is expressed by

$$y^+ = \frac{\sqrt{Re\,\rho_w \tau_w}\, y}{\mu_w} \tag{2.3.5}$$

in which ρ_w, μ_w are the density and the molecular coefficient of viscosity at the wall, respectively, and τ_w is the magnitude of the wall shear stress.

For the viscosity of outer layer, $(\mu_t)_{outer}$ is given by

$$(\mu_t)_{outer} = Re\, KC_{cp} \rho \Gamma_{wake} F_{kleb}(y) \tag{2.3.6}$$

where K is the Clauser constant, Ccp is an additional constant, and

$$\Gamma_{wake} = \min \text{ of } \left\{ \begin{matrix} y_{max}\Gamma_{max} \\ C_{wk}\, y_{max} \dfrac{U_d^2}{\Gamma_{max}} \end{matrix} \right\} \tag{2.3.7}$$

The quantities y_{max} and Γ_{max} are determined from the function

$$\Gamma(y) = y|\omega|\left[1 - \exp\left(-y^+ / A^+\right)\right] \tag{2.3.8}$$

The quantity Γ_{max} is the maximum value of $\Gamma(y)$ and y_{max} is the value of y at which $\Gamma(y)$ reaches Γ_{max}. The function $F_{kleb}(y)$ is the Klebanoff intermittency factor given by

$$F_{Kleb}(y) = \left[1 + 5.5\left(\frac{C_{Kleb}\, y}{y_{max}}\right)^6\right]^{-1} \tag{2.3.9}$$

Lastly, U_d is defined as

$$U_d = \left(\sqrt{u^2 + v^2 + w^2}\right)_{max} - \left(\sqrt{u^2 + v^2 + w^2}\right)_{min} \tag{2.3.10}$$

where the subscripts, max and min, refer to the maximum and minimum values of a given profile. The minimum value is always zero except in wakes. Additionally, the exponential term in (2.3.8) is set to zero in wakes.

The effect of transition to turbulence can be specified at a point or can be simulated by setting the turbulent viscosity to zero everywhere in the profile unless the maximum value of it in the profile computed from the above formulations exceeds some threshold value. Baldwin and Lomax suggested that this threshold value be set to $14\mu_{mol}$, where μ_{mol} is the molecular viscosity.

The values of the constants are given as follows

$A^+=26$ $C_{cp}=1.6$ $C_{Kleb}=0.3$ $K=0.0168$ $k=0.4$.

2.4 Degani and Schiff Modification of B-L Model

A problem with Baldwin-Lomax model is encountered when it is applied to bodies with crossflow separation. In these separated flow regions, it becomes difficult to determine the correct values for y_{max} and Γ_{max}, which, in turn, produces erroneous results for $(\mu_t)_{outer}$. This problem is because the behaviour of the function $\Gamma(y)$ in a region of crossflow separation which, normally has one maximum value in a region of attached flow, has two relative maxima. The first peak occurs within the boundary layer, and a second, larger peak exists due to the presence of the vortex sheet. If Baldwin-Lomax model is used to search outward along each ray to determine the maximum in $\Gamma(y)$, it would select the second peak. This results in values of Γ_{max}, y_{max} and the resulting value of $(\mu_t)_{outer}$ are much too high, and results in a distortion or a washout of the features in the computed flow.

To address this problem, Degani and Schiff[4] proposed a modification to the application of Baldwin-Lomax model. Instead of searching outward along the entire radial ray for the value of Γ_{max}, a criterion is established to differentiate between the two local maxima and to limit the search outward along the ray. The search is cut off when the first peak is reached, and in order to prevent the selection of extraneous peaks which might be caused by a nonsmooth behaviour in $\Gamma(y)$, a peak is considered to have been found when the value of $\Gamma(y)$ drops to 90% of the local maximum value.

For most rays in the region of crossflow separation the two peaks are spaced far enough apart, but along the rays near crossflow separation point, the two peaks tend to merge since the overlying vortex structure is close to the boundary layer, and no definite peak may occur. In this case, Degani and Schiff suggested a cut-off distance which is specified in terms of y_{max} from the adjoining, more windward ray, that is, $y_{cutoff}(\phi)=1.5y_{max}(\phi-\Delta\phi)$ where ϕ is the angle of a ray from the windward side. If no peak in $\Gamma(y)$ is found along a ray for $y \le y_{cutoff}$ the values of Γ_{max} and y_{max} are taken as those found on the adjoining, more windward, ray.

The cutoff distance for searching the desired values of Γ_{max} and y_{max} is shown schematically in Fig. 2.4.1.

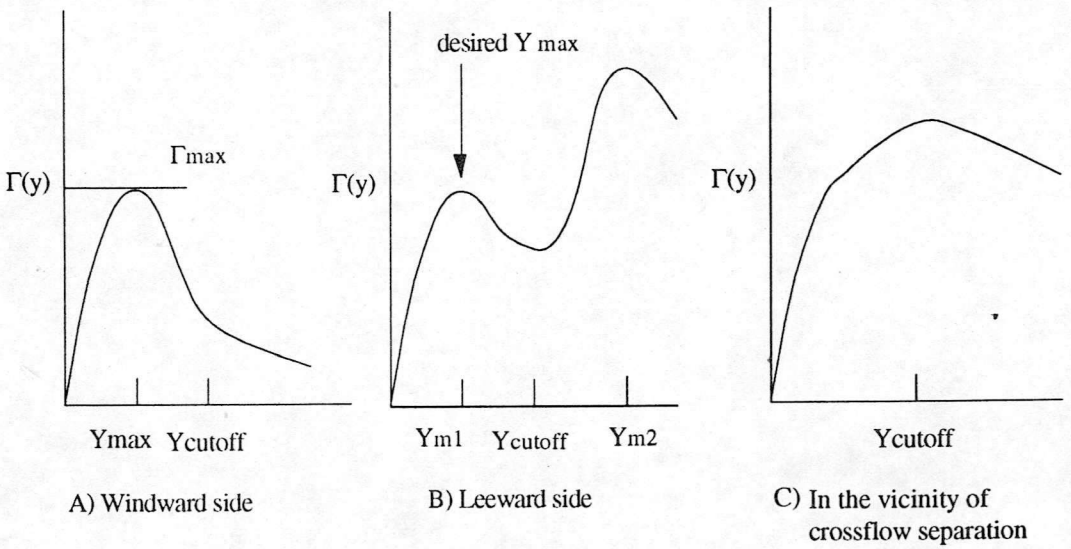


Figure 2.4.1 Behaviour of $\Gamma(y)$ in crossflow of a slender body.

2.5 Johnson-King Turbulence Model

Another turbulence model coded in this program is one developed by Johnson and King[5]. The developers characterize this turbulence model as a half-equation, Reynolds stress, eddy-viscosity model, in which an ordinary differential equation (ODE) is used to determine the maximum Reynolds shear stress, rather than the partial differential equation used in one-equation models. The closure model is composed of a nonequilibrium eddy-viscosity distribution, a rate equation for the streamwise development of the Reynolds shear stress, and an equilibrium eddy-viscosity distribution, in which production equals dissipation.

As with the Baldwin-Lomax model, the Johnson-King model divides the boundary layer into an inner and outer region. The turbulent eddy viscosity μ_t is a function of the viscosity values for these two regions and is assumed to be

$$\mu_t = (\mu_t)_{outer}\left\{1-\exp\left[-(\mu_t)_{inner}/(\mu_t)_{outer}\right]\right\} \tag{2.5.1}$$

This equation provides a smooth blending between the inner and outer regions of the boundary layer and makes μ_t dependent on $(\mu_t)_{outer}$ across most of the boundary layer.

The inner eddy viscosity is given by

$$(\mu_t)_{inner} = \rho\,Re\,D^2 ky\sqrt{\tau_m} \tag{2.5.2}$$

where τ_m is the maximum Reynolds shear stress, y is the local normal distance from the wall, k is the von Karman constant, and the damping term D is defined as

$$D = 1-\exp\left(\frac{-\rho_w u_T\sqrt{Re}\,y}{\mu_w A^+}\right) \tag{2.5.3}$$

where A^+ is a constant and u_T is defined as

$$u_T = \max\left(\sqrt{\tau_m/\rho_m},\sqrt{\tau_w/\rho_w}\right) \tag{2.5.4}$$

where τ_w is the wall shear stress.

The Reynolds shear stress is assumed to be

$$\tau = \mu_t\Omega/\rho \tag{2.5.5}$$

where Ω is the magnitude of the vorticity.

The outer eddy viscosity is determined by using the Baldwin-Lomax wake model with a parameter σ

$$(\mu_t)_{outer} = \sigma\,Re\,KC_{cp}\rho\Gamma_{wake}F_{kleb}(y) \tag{2.5.6}$$

where the functions and parameters are the same as those provided for Baldwin-Lomax model except that $\Gamma(y)$ is composed of the following three factors instead of (2.3.7)

$$\Gamma(y) = y|\omega|D \tag{2.5.7}$$

The unknown parameter σ provides the link between eddy viscosity distribution of equation (2.5.1) and the rate equation for the streamwise development of the maximum Reynolds shear stress.

The τ_m is obtained from the solution of a partial differential equation in three dimensional space. Although this three-dimensional PDE was extended by Abid et al[6] based on the Cartesian coordinate system, there are some confusable concepts which must be cleared before it is introduced. Initially, this rate equation was derived for two-dimensional incompressible flow by Johnson and King in such a local coordinate system that the normal outward direction from the wall and the streamwise direction were the two coordinate axes, though the Cartesian coordinates were used for the approximation. The unknown function in the original ordinary differential equation was the maximum Reynolds shear stress along each radial ray from the wall. The maximum Reynolds shear stress, though the value itself and its location vary, is a single value along each radial ray. Therefore the equation was for a local variable rather than a whole field variable, such as the Reynolds shear stress itself. A partial differential equation across the whole field for τ_m may be misleading for implementation of the model.

In order to give clearer background, the streamline coordinate system (s,n) is chosen, and the rate equation based on this coordinate system can be written as

$$U_m \frac{d\tau_m}{ds} = \frac{a_1}{L_m}\tau_m\left[\sqrt{\tau_{m,eq}} - \sqrt{\tau_m}\right] - a_1 D_m \qquad (2.5.8)$$

where U_m is the magnitude of the velocity at the location of the maximum Reynolds shear stress, L_m is the dissipation length scale, D_m the turbulent diffusion term, the expressions of these two are given as follows

$$L_m = \min(0.4y_m, 0.09\delta) \qquad (2.5.9)$$

$$D_m = \frac{C_D \tau_m^{3/2}\left|1-\sqrt{\sigma}\right|}{a_1(0.7\delta - y_m)} \qquad (2.5.10)$$

where y_m is the normal distance from the surface to the location of the maximum Reynolds shear stress, and δ the boundary-layer thickness which is equal to $1.9y_{max}$ (which is the same y_{max} used in Baldwin-Lomax model).

Making the following change in variables

$$g = 1/\sqrt{\tau_m}; \quad g_{eq} = 1/\sqrt{\tau_{m,eq}} \qquad (2.5.11)$$

equation (2.5.8) can be rewritten as follow

$$U_m \frac{dg}{ds} = \frac{a_1}{2L_m}\left(1 - \frac{g}{g_{eq}}\right) + D'_m \qquad (2.5.12)$$

where D'_m is given as

$$D'_m = \frac{C_D\left|1-\sqrt{\sigma}\right|}{2(0.7\delta - y_m)} \qquad (2.5.13)$$

The parameter $\sigma(s)$ is used for linking the maximum shear stress value as determined from the rate equation and the maximum shear stress value as determined from the eddy viscosity distribution of the flow-filed. Both values are used for updating $\sigma(s)$ at the next time level according to the following relation:

$$\sigma^{t+dt} = \sigma^t \tau_m \left(\frac{\rho}{\mu_t \Omega} \right)_{max} \tag{2.5.14}$$

The initial conditions for the rate equation are taken to be those found in an equilibrium eddy-viscosity turbulent model, such as the Baldwin-Lomax model, that is,

$$\mu_t = \mu_{t,eq} \quad g = g_{eq} \tag{2.5.15}$$

The values of the constants are given (if different from those given for Baldwin-Lomax model) as follows

$A^+=17$ $a_1=0.25$ $C_D=0.5$

Following reference [7], the solution of the rate equation can be worked out through the following procedures:

1). At each surface grid point, the ray normal to the body is searched to find τ_m at first using the shear stress distribution (2.5.5), and after solving the equation τ_m is found from the previous iteration step. In that formula the eddy-viscosity value is from the previous time step.

2). Calculating the equilibrium shear stress value $\tau_{m,eq}$ in (2.5.11) using the following formula for the inner viscosity:

$$\left(\mu_{t,i}\right)_{eq} = \rho D^2 k y \sqrt{\rho_m/\rho}\, u_{m,eq} \tag{2.5.16}$$

and setting $\sigma=1$ in the equation for the outer viscosity (2.5.6). The equilibrium turbulent eddy-viscosity coefficient is then determined using equation (2.5.1).

3). Knowing the velocity components of the flow at the location of τ_m along the ray, it is possible to calculate the direction of the flow and, thus, the direction of the shear stress direction.

4). The rate equation is then solved by using Taylor series method or Runge-Kutta method. The right-hand-side of the equation is determined by using the data from the previous time step.

5). Knowing τ_m from the ODE, the linking parameter σ is then updated using relation (2.5.14).

Some questions for implementing this model still remain: where is the starting location of the equilibrium eddy-viscosity for each ray, and how far this streamline should go, and how to follow the streamline in three dimensional grid?

An alternative is to solve the following three-dimensional partial differential equation[6]

$$\frac{\partial g}{\partial t} + u_m \frac{\partial g}{\partial x} + v_m \frac{\partial g}{\partial y} + w_m \frac{\partial g}{\partial z} = \frac{a_1}{2L_m}\left(1 - \frac{g}{g_{eq}}\right) + D_m' \tag{2.5.17}$$

where u_m,v_m and w_m are the velocity components at the location of maximum Reynolds shear stress. Although this PDE is only valid for the maximum Reynolds shear stress, it can be used for the whole field and then along each ray the maximum value of the solution is searched as the real maximum Reynolds shear stress.

Following reference [6], solutions based on this model are obtained as follows:

1). A steady-state solution is obtained by using the equilibrium model, that is, the right-hand-side is set up equal to zero for the calculation.

2). After 35 cycles the model in activated.

3). At each time advance, two values of the maximum shear stress are determined; one based on the maximum value using (2.5.5), and the other τ_m based on the integration of equation (2.5.17). These two are then used to update σ using relation (2.5.14).

3.NUMERICAL ALGORITHM

3.1 Explicit Finite Volume Method

The Navier-Stokes equations can be solved by different methods. The finite volume method is preferred to the finite difference method because of its superior conservative property. The explicit finite volume method, which is the easiest to adopt, stems from the time-dependent form of the equations, which naturally reaches the steady-state when the time interval is long enough. If we start from the full set of Navier-Stokes equations (2.1.1), the integral form of them can be written as

$$\int_{\Omega}\left(\frac{\partial Q}{\partial t}+\frac{\partial(E_i-E_v)}{\partial x}+\frac{\partial(F_i-F_v)}{\partial y}+\frac{\partial(G_i-G_v)}{\partial z}\right)d\Omega=0 \qquad (3.1.1)$$

where Ω is the domain of a control volume. Integrating them by parts yields

$$\int_{\Omega}\frac{\partial Q}{\partial t}d\Omega+\int_{S}(E_i-E_v)ds_x+(F_i-F_v)ds_y+(G_i-G_v)ds_z=0 \qquad (3.1.2)$$

where S is the boundary of the domain Ω. If the domain of a problem is broken into finite number of cells, the integral equations (3.1.2) can be rewritten for each of the cells, that is,

$$\int_{\Omega_k}\frac{\partial Q}{\partial t}d\Omega+\int_{S_k}(E_i-E_v)ds_x+(F_i-F_v)ds_y+(G_i-G_v)ds_z=0 \qquad (3.1.3)$$

and the whole solution of the problem can then be worked by simultaneously solving the equations for all the cells. For each cell, such average state of the conservative hypervector Q over the cell is taken from two adjacent time steps that the first integral can become

$$\int_{\Omega_k}\frac{\partial Q}{\partial t}d\Omega=\frac{V_k}{\Delta t}\left(\overline{Q(t+\Delta t)}-\overline{Q(t)}\right)_k \qquad (3.1.4)$$

where V_k is the volume of cell Ω_k, and $\overline{Q(t)_k}$ correspond to the average state of the conservative hypervector over the cell at the moment t. The integral conservation relation (3.1.3) becomes

$$\overline{Q_k(t+\Delta t)}=\overline{Q(t)_k}-\frac{\Delta t}{V_k}\int_{S_k}(E_i-E_v)ds_x+(F_i-F_v)ds_y+(G_i-G_v)ds_z \quad (3.1.5)$$

This is an exact relation of the conservation form, and is the basis of the explicit finite volume method used in the solver.

If each cell is formed by the local coordinates of a structured mesh, and the area vectors of these sides are given by

$$d\vec{\xi}=(d\xi_x, d\xi_y, d\xi_z);\ d\vec{\eta}=(d\eta_x, d\eta_y, d\eta_z);\ d\vec{\zeta}=(d\zeta_x, d\zeta_y, d\zeta_z) \quad (3.1.6)$$

and equations (3.1.5) can then be approximated by the following finite volume expressions:

$$\begin{aligned}\overline{Q_k(t+\Delta t)}=\overline{Q(t)_k}-\frac{\Delta t}{V_k}\Big\{&\left[(E_i-E_v)d\xi_x+(F_i-F_v)d\xi_y+(G_i-G_v)d\xi_z\right]_-^+\\&+\left[(E_i-E_v)d\eta_x+(F_i-F_v)d\eta_y+(G_i-G_v)d\eta_z\right]_-^+\\&+\left[(E_i-E_v)d\zeta_x+(F_i-F_v)d\zeta_y+(G_i-G_v)d\zeta_z\right]_-^+\Big\}\end{aligned} \quad (3.1.7)$$

where the convective and the diffusive fluxes are at the corresponding boundaries, and +, - signs correspond the entry side and exit side of the cell. This column hypervector expression shows that the variation, over the time interval Δt, of the cell-averaged state of the conservative hypervector Q as resulting from the balance of the time-averaged fluxes at the boundaries of the cell.

3.2 Osher's Approximate Riemann Solver

From above finite volume formulations it can be seen that the calculation of the fluxes at the cell boundaries is crucial to the solution of the problem, and especially for supersonic and/or hypersonic flows, a family of upwind schemes for the inviscid fluxes evaluations have been constructed. Among those schemes Osher's approximate Riemann solver appears to be the best choice for the problem.

Consider the Riemann problem for the corresponding Euler equations. The Osher's approximate Riemann solver is given by

$$\vec{F}_i(Q^L, Q^R)=\frac{1}{2}\left[\vec{F}_i(Q^L)+\vec{F}_i(Q^R)-\int_{Q^L}^{Q^R}\left|\frac{\partial\vec{F}_i}{\partial Q}\right|dQ\right] \quad (3.2.1)$$

where $\vec{F}_i$ is the compact form of the column hypervectors, or hyper-tensor, of the inviscid fluxes through a boundary with given variables Q^L on the left and Q^R on the right of it.

The evaluation of the integrals relies on the Riemann invariants along the local area vector direction, and the integral path can then be composed by three sections which is depicted in the

following figure. The path can actually correspond with two variants: P-variant and O-variant [see Spekreijse Ref.8]. P-variant is more natural and easier to adopt so we chose it for the algorithm here.

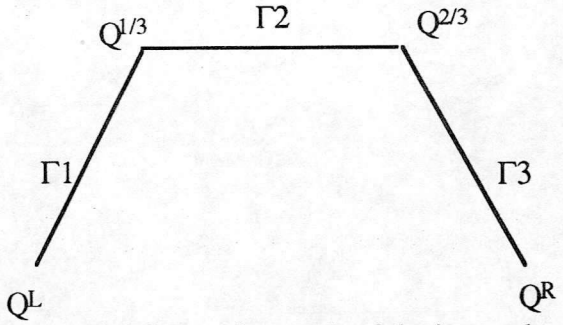


Fig. 3.2.1 P-variant path of the integral

From the conservative variables Q^L and Q^R we can obtain the information of $Q^{1/3}$ and $Q^{2/3}$ together with the local sound speeds, therefore the fluxes evaluations become straightforward. The result is summarised in the following table.

	$u_L<c_L, u_R>-c_R$	$u_L>c_L$, $u_R>-c_R$	$u_L<c_L$, $u_R<-c_R$	$u_L>c_L$, $u_R<-c_R$
$c_{1/3}< u_H$	$\vec{F}_s^{\,L}$	$\vec{F}^{\,L}$	$\vec{F}_s^{\,L}-\vec{F}_s^{\,R}+\vec{F}^{\,R}$	$\vec{F}^{\,L}-\vec{F}_s^{\,R}+\vec{F}^{\,R}$
$0<u_H<c_{1/3}$	$\vec{F}^{\frac{1}{3}}$	$\vec{F}^{\,L}-\vec{F}_s^{\,L}+\vec{F}^{\frac{1}{3}}$	$\vec{F}^{\frac{1}{3}}-\vec{F}_s^{\,R}+\vec{F}^{\,R}$	$\vec{F}^{\,L}-\vec{F}_s^{\,L}+\vec{F}^{\frac{1}{3}}-\vec{F}_s^{\,R}+\vec{F}^{\,R}$
$-c_{1/3}<u_H<0$	$\vec{F}^{\frac{2}{3}}$	$\vec{F}^{\,L}-\vec{F}_s^{\,L}+\vec{F}^{\frac{2}{3}}$	$\vec{F}^{\frac{2}{3}}-\vec{F}_s^{\,R}+\vec{F}^{\,R}$	$\vec{F}^{\,L}-\vec{F}_s^{\,L}+\vec{F}^{\frac{2}{3}}-\vec{F}_s^{\,R}+\vec{F}^{\,R}$
$u_H<-c_{1/3}$	$\vec{F}_s^{\,R}$	$\vec{F}^{\,L}-\vec{F}_s^{\,L}+\vec{F}_s^{\,R}$	$\vec{F}^{\,R}$	$\vec{F}^{\,L}-\vec{F}_s^{\,L}+\vec{F}^{\,R}$

where the superindex indicates that the flux is calculated by the local variables Q.

3.3 The MUSCL Scheme

The left and right conservative variables are determined by interpolations which determine the accuracy of the scheme. In order to give higher order accuracy, as well as to avoid non-physical states such as negative pressure, the primitive variables (ρ, u, v, w, T) are actually used, and a κ-parameter family scheme[9] is adopted for the interpolations,e.g. for E-wave:

$$\left.\begin{aligned} V^L &= V_{i,j,k}+\left\{\left(\frac{s}{4}\right)\left[(1-\kappa s)\Delta_- +(1+\kappa s)\Delta_+\right]V\right\}_{i,j,k} \\ V^R &= V_{i+1,j,k}-\left\{\left(\frac{s}{4}\right)\left[(1+\kappa s)\Delta_- +(1-\kappa s)\Delta_+\right]V\right\}_{i+1,j,k} \end{aligned}\right\} \qquad (3.3.1)$$

where Δ_+ and Δ_- denote forward and backward difference operators, respectively, in η direction; and V is the column hypervector of the primitive variables. The parameter κ determines the spatial accuracy of the interpolation since κ=-1 corresponds to a fully-upwind second order scheme, κ=1 to a central difference scheme, and κ=1/3 to a third order upwind-biased scheme. The parameter s serves to limit higher-order terms in the interpolation in order to avoid oscillations at discontinuities such as shock waves in the solutions. According to

Anderson et al[10], the limiting is implemented by locally modifying the difference values in the interpolation to ensure monotone interpolation. s expression is given as

$$s = \frac{2\Delta_+ V \Delta_- V + \varepsilon}{(\Delta_+ V)^2 + (\Delta_- V)^2 + \varepsilon} \tag{3.3.2}$$

where ε is a small number preventing division by zero in regions of null gradients.

3.4 Evaluation of Diffusive Fluxes

For the evaluation of the diffusive fluxes it is necessary to approximate the gradients ∇u, ∇v, ∇w, ∇T at each boundary of all the cells. The Gauss theorem can be used here, for example,

$$\nabla u = \frac{1}{V} \int_S u \, d\vec{S} \tag{3.4.1}$$

However the integral domain for this integral at each side of a cell is no longer the cell boundary itself. Therefore an offset boundary around the side centre is used and both the area vectors and the variables on this boundary have to be approximated by the values from both sides. For instance to evaluate a derivative at the $d\eta_{i+1/2,j,k}$ which refers to the cell interface between cell (i,j,k) and (i+1,j,k), the following approximation is used

$$\left(\frac{\partial u}{\partial x}\right)_{i+1/2,j,k} = \left(u^F d\xi_x^F - u^B d\xi_x^B + u^R d\eta_x^R - u^L d\eta_x^L + u^U d\zeta_x^U - u^D d\zeta_x^D\right) \Big/ V_{i+1/2,j,k} \tag{3.4.2}$$

where $V_{i+1/2,j,k}$ is the average volume of cell i,j,k and i+1,j,k, and

$$\left.\begin{aligned} u^R &= \left(u_{i,j+1,k} + u_{i+1,j+1,k} + u_{i,j,k} + u_{i+1,j,k}\right)/4 \;; \quad u^F = u_{i+1,j,k} \\ u^L &= \left(u_{i,j-1,k} + u_{i+1,j-1,k} + u_{i,j,k} + u_{i+1,j,k}\right)/4 \;; \quad u^B = u_{i,j,k} \\ u^U &= \left(u_{i,j,k+1} + u_{i+1,j,k+1} + u_{i,j,k} + u_{i+1,j,k}\right)/4 \\ u^D &= \left(u_{i,j,k-1} + u_{i+1,j,k-1} + u_{i,j,k} + u_{i+1,j,k}\right)/4 \end{aligned}\right\} \tag{3.4.3}$$

$$\left.\begin{aligned} d\xi_x^F &= 0.5\left(d\xi_x^{i,j,k} + d\xi_x^{i+1,j,k}\right) \;; \quad d\xi_x^B = 0.5\left(d\xi_x^{i,j,k} + d\xi_x^{i-1,j,k}\right) \\ d\eta_x^R &= 0.5\left(d\eta_x^{i,j,k} + d\eta_x^{i+1,j,k}\right) \;; \quad d\eta_x^L = 0.5\left(d\eta_x^{i,j-1,k} + d\eta_x^{i+1,j-1,k}\right) \\ d\zeta_x^U &= 0.5\left(d\zeta_x^{i,j.k} + d\zeta_x^{i+1.j,k}\right) \;; \quad d\zeta_x^D = 0.5\left(d\zeta_x^{i,j.k-1} + d\zeta_x^{i+1,j,k-1}\right) \end{aligned}\right\} \tag{3.4.4}$$

which formulas come from a finite difference method.

Similar expressions are used for the other gradients and other interfaces of the cell.

3.5 CFL Condition for Time Stepping

Since the time-dependent approach is used to solve the steady-state equations, the convergence depends on the local time step which has to be determined by the CFL condition for each cell. The formulation for the time step of the numerical NS equations is given as follow:

$$\Delta t \le \frac{CFL}{\left|\vec{u}\bullet d\vec{\eta}\right| + \left|\vec{u}\bullet d\vec{\xi}\right| + \left|\vec{u}\bullet d\vec{\zeta}\right| + c\left(\left|d\vec{\eta}\right| + \left|d\vec{\xi}\right| + \left|d\vec{\zeta}\right|\right) + a\left(\left|d\vec{\eta}\right|^2 + \left|d\vec{\xi}\right|^2 + \left|d\vec{\zeta}\right|^2\right)} \qquad (3.5.1)$$

in which c is the local sound speed, CFL a prescribed constant and $a = \frac{2\gamma\mu}{\rho P_r Re}$.

3.6 Boundary Conditions

There are four kinds of boundary conditions used in this solver: inflow, outflow, solid wall and symmetrical plane.

The inflow condition is to provide the flow field information of the previous station.

The outflow condition corresponds the flow field of the next station, since the flow is mainly supersonic, no boundary condition is needed.

On the solid wall, no-slip conditions are applied together with the isothermal wall condition. The pressure gradient at the wall is zero.

For evaluation of the fluxes at a symmetrical plane, a mirror cell is used to reflect the symmetric and unsymmetrical variables. Thus the normal procedure can be used.

4.CODE DESCRIPTION

A complete listing of the code NS3DT is given in what follows, together with a description of each subroutine and the files included. Data and resulting post-processed pictures of two test cases will be presented in the next section. The program is coded in FORTRAN 77 and is fully self-contained. It can be easily ported and has been running successfully on IBM-3090/150E(CMS), IBM-RISC/6000-320H, Silicon Graphics-IRIS/ INDIGO, SUN SPARC-10, Intel iPSC/860 Parallel Computer and on RAL's Cray YMP. The efficiency and readability of the code have been improved much more than its original base code[1], with the memory requirement being reduced.

4.1 Main program

The main program is listed first hereafter:

```
C       PROGRAM NS3DT
        INCLUDE "comm.f"
C
        CALL INP3D
        CALL GRI3D
        CALL INI3D
        IT=-1
2000    IT=IT+1
        CALL BFL3D
        CALL ROS3D
        CALL UPD3D
        IF(IT.GE.INTVAR)THEN
         IF(ITYPE.EQ.1)CALL BLT3D
         IF(ITYPE.EQ.2)THEN
          IF(IT.LT.INTVAR+35)THEN
           CALL BLT3D
          ELSE
           CALL JKT3D
           CALL RSS3D
          ENDIF
         ENDIF
        ENDIF
        IF(R2NORM.GT.CRIIT.AND.IT.LE.ITMAX)GOTO 2000
        CALL OUT3D
        CALL HTC3D
        CLOSE(8)
        DO 4000 I=1,7
        CLOSE(9+I)
4000    CLOSE(20+I)
        STOP
        END
```

This main part of the program can be read as the main flow-chart, because it shows the sequential order of the major modules. The first module is INP3D which inputs the controlling data from the data file, and put them and other parameters calculated from them into the memory. GRI3D reads the grid information of the problem and calculates the area vector components and volume of each cell. INI3D initialises the flow field of the first station. Iteration begins at label 2000 where the counter IT is added one more each time. The core of the program starts from BFL3D in which the boundary fluxes are calculated. It is followed by ROS3D in which the fluxes between each adjacent cells inside the problem domain are computed and the residuals of all the cells are collected. The flow field variables are then updated in UPD3D and the second type of norm of the residuals is also summed there. The

turbulence models are optional, depending on the input parameters INTVAR and ITYPE, where INTVAR controls the starting iteration number for turbulence modelling, whereas ITYPE gives the model type. The Baldwin-Lomax model and Johnson-King model are coded in BLT3D and JKT3D, respectively. The solution of the "half-equation" for the maximum Reynolds shear stress is performed along with the main solver for the Navier-Stokes equations. The iteration converges when the norm is not bigger than a prescribed parameter CRIIT, or it is forced to end when the ITMAX is reached. OUT3D is mainly for outputing the result. An additional subroutine HTC3D is for outputing the heat transfer around the wall.

4.2 COMMON Declarations

From the CALL statements for the modules it can be readily seen that the data transmition between subroutines is mainly done through COMMON blocks. The declaration of the COMMON blocks are stored in a file called "comm.f" which is included in the first line in the main program, also included in the beginning of every major subroutine. The listing and the explanation of this included file are given hereafter.

```
      PARAMETER (IM=56,JM=66,KM=76)
      COMMON /PHYPA/ XMINF,TINF,TW,RE,PR,GAM,ALPHA,TINFA,RTINFA,RRE
      COMMON /GAMS / GAMC0,GAMC1,GAMC2,GAMC3,GAMC4,GAMC5,GAMC6,C0,CT
      COMMON /NUMPA/ CFL,CRIIT,EPS,CRIT1,TIM1,TIME,UDIF
      COMMON /LIMIT/ IN,JN,KN,ITMAX,IRST,IT,R1NORM,R2NORM,R3NORM
      COMMON /CONTL/ IORDER,ITYPE,CK,PHI,INTVAR,ISHARP
      COMMON /MESH / SXI(IM,JM,KM,3),SETA(IM,JM,KM,3),
     &               SZET(IM,JM,KM,3),VOL(IM,JM,KM),AREA(IM,JM,KM,3)
      COMMON /RESID/ RR(IM,JM,KM,5)
      COMMON /TURBU/ DS(IM,JM,KM),XMUT(IM,JM,KM)
```

The first five COMMONs are controlling parameters which are read or calculated in INPPNS module. The first of them is mainly the physical parameters, in which XMINF represents for M_∞, TINF for T_∞, TW for temperature at the wall, RE for Reynolds number, PR for Prandtl number P_r, GAM for γ, ALPHA for angle of attack α, and the rest of the parameters are generated in INPPNS. COMMON /GAMS/ mainly provides the γ-related parameters to be used in the program. The third block contains control parameters for numerics in which CFL is the constant for CFL condition, CRIIT the criteron for convergence, EPS is the limiter constant ε, CRIT1 and UDIF are used for working spaces. COMMON /LIMIT/ supplies the mesh and iteration bounds such as IN, JN, KN are the maximum numbers of the grid points in I, J, K directions of a structured mesh, ITMAX is the maximum iteration times. The fifth block has the controlling parameters like the order of the κ-parameter family scheme (IORDER), flow type (ITYPE a vacant parameter), INTVAR the Ith station to switch to turbulent flow, and I0 which is the I-number of a mesh for the first station to start with. ISHARP is a parameter to give the shape of the nose, ISHARP=0 for blunt nose and 1 for sharp nose.

All the arrays are defined based on IM, JM & KM which are the maximum values of I, J and K, respectively. These three parameters define the requirement of the problem for computer storage, since the total size is a little bit more than 26xIMxJMxKM. The representations of the arrays are given as follows:

SXI(IM,JM,KM,3)	Three components of $d\xi$ of each cell
SETA(IM,JM,KM,3)	Three components of $d\eta$ of each cell
SZET(IM,JM,KM,3)	Three components of $d\zeta$ of each cell
AREA(IM,JM,KM,3)	Three areas of the area vectors of each cell
VOL(IM,JM,KM)	Volume of each cell
RHO(IM,JM,KM)	Density of each cell

U(IM,JM,KM)	Velocity x-component of each cell
V(IM,JM,KM)	Velocity y-component of each cell
W(IM,JM,KM)	Velocity z-component of each cell
T(IM,JM,KM)	Temperature of each cell
XMU(IM,JM,KM)	Molecular viscosity of each cell
RR(IM,JM,KM,5)	Residual of integration of each cell
DS(IM,JM,KM)	Normal distance from the wall of each cell
XMUT(IM,JM,KM)	Turbulent viscosity of each cell

RHO,U,V,W,T,XMU are declared in BFL3D, INI3D, ROS3D, UPD3D and OUT3D, the space is also used for storing the coordinates temporarily in GRI3D.

4.3 Subroutine INP3D

This subroutine is designed to deal with the controlling parameters which are read into the program from the data file called "inp3dt.data". In order to distinguish the descriptions of the data and themselves, some header character variables are used. The grid file name is also a parameter in the data, which is used to open the file for mesh input. The starting station's number I0 is read into the program just before the file is closed. Parameter GAM (γ) has been used in its various expressions, a common block called GAMS is used and the parameters of it are all generated here. RRE is the inverse of Reynolds number, C0 and CT are two parameters to serve for the temperature diffusive fluxes of laminar and turbulent terms.

```
      SUBROUTINE INP3D
      INCLUDE "comm.f"
      CHARACTER HL0*80,HL1*80,HL2*80,HL3*80,HL4*80
C
C  Read in file header.
C
      OPEN(15,FILE='inp3dt.data',FORM='FORMATTED',STATUS='OLD')
C
      READ(15,100) HL0
      READ(15,100) HL1
      READ(15,*) IORDER,ITYPE,INTVAR,ISHARP
      READ(15,100) HL2
      READ(15,*) ALPHA,CFL,CRIIT,EPS,R1NORM
      READ(15,100) HL3
      READ(15,*) XMINF,TINF,TW,RE,PR,GAM
      READ(15,100) HL4
      READ(15,*) IN,JN,KN,ITMAX,IRST
C
      OPEN(16,FILE='check.data',FORM='FORMATTED',STATUS='UNKNOWN')
C
      WRITE(16,100) HL0
      WRITE(16,100) HL1
      WRITE(16,150) IORDER,ITYPE,INTVAR,ISHARP
      WRITE(16,100) HL2
      WRITE(16,200) ALPHA,CFL,CRIIT,EPS,R1NORM
      WRITE(16,100) HL3
      WRITE(16,400) XMINF,TINF,TW,RE,PR,GAM
      WRITE(16,100) HL4
      WRITE(16,300) IN,JN,KN,ITMAX,IRST
      ALPHA=ALPHA*3.1415927/180.0
C
C---- PARAMETERS FOR GAMS ----------
```

```
C
      EPS=EPS*EPS
      GAMC0=2.0/(GAM-1.0)
      GAMC1=0.5/GAM
      GAMC2=GAM/(GAM-1.0)
      GAMC3=1.0/GAM
      GAMC4=1.0/(GAM-1.0)
      GAMC5=(GAM-1.0)/(GAM+1.0)
      GAMC6=1.0/(GAM*XMINF**2)
      TINFA=TINF/120.0
      RTINFA=1.0/TINFA
      RRE=1.0/RE
      C0=0.5*RRE/((GAM-1.0)*XMINF**2*PR)
      CT=0.5*RRE/((GAM-1.0)*XMINF**2*0.9)
      CK=0.0
      PHI=1.0
      IF(IORDER.EQ.1)PHI=0.0
      IF(IORDER.EQ.2)CK=-1.
      IF(IORDER.EQ.3)CK=1.0/3.0
      TIM1=0.0
C
 100  FORMAT(A80)
 150  FORMAT(4I10)
 200  FORMAT(2F10.5,3E10.3)
 300  FORMAT(5I10)
 400  FORMAT(3F10.5,E10.3,2F10.5)
      RETURN
      END
```

4.4 Subroutine GRI3D

This subroutine is for reading the grid coordinates to generate the area vector components and volumes. Subroutine AXB is called for multiplication of two three-component vectors, the list of which is given in the end of this subroutine. The last arguement of AXB is the area (or the magnitude) of the vector produced. Array AERA is used to store the areas of all the cells in order to increase the speed of computation. For turbulent modelling, the normal distances from the wall for all the centroids of the cells are calculated here, and the coordinates will not be used in the program until the next station is read in. The common block /STATE/ is used temporarily here since the coordinates are once for all, and the store space for the primitive variables will be used after this subroutine is called.

```
      SUBROUTINE GRI3D
      INCLUDE "comm.f"
      COMMON /STATE/ CO(IM,JM,KM,3)
      DIMENSION CO1(3),CO2(3),CO3(3)
      CHARACTER*19 NAME
      READ(15,15)NAME
15    FORMAT(A19)
      CLOSE(15)
      OPEN(10,FORM='UNFORMATTED',FILE=NAME)
      READ(10)(((CO(I,J,K,1),I=1,IN),J=1,JN),K=1,KN)
      READ(10)(((CO(I,J,K,2),I=1,IN),J=1,JN),K=1,KN)
      READ(10)(((CO(I,J,K,3),I=1,IN),J=1,JN),K=1,KN)
      CLOSE(10)
      DO 50 K=1,KN
      DO 50 J=1,JN
      DO 50 I=1,IN
```

```
 50     CO(I,J,K,1)=-CO(I,J,K,1)
        DO 100 K=2,KN
        DO 100 J=2,JN
        DO 100 I=1,IN
        DO 110 L=1,3
        CO1(L)=CO(I,J,K,L)-CO(I,J-1,K-1,L)
        CO2(L)=CO(I,J,K-1,L)-CO(I,J-1,K,L)
 110    CONTINUE
        CALL AXB(CO1,CO2,CO3)
        AREA(I,J,K,1)=0.5*SQRT(CO3(1)**2+CO3(2)**2+CO3(3)**2)
        DO 120 L=1,3
 120    SXI(I,J,K,L)=0.5*CO3(L)
  100   CONTINUE
        DO 200 K=2,KN
        DO 200 J=1,JN
        DO 200 I=2,IN
        DO 210 L=1,3
        CO1(L)=CO(I,J,K,L)-CO(I-1,J,K-1,L)
 210    CO2(L)=CO(I-1,J,K,L)-CO(I,J,K-1,L)
        CALL AXB(CO1,CO2,CO3)
        AREA(I,J,K,2)=0.5*SQRT(CO3(1)**2+CO3(2)**2+CO3(3)**2)
        DO 220 L=1,3
 220    SETA(I,J,K,L)=0.5*CO3(L)
  200   CONTINUE
        DO 300 K=1,KN
        DO 300 J=2,JN
        DO 300 I=2,IN
        DO 310 L=1,3
        CO1(L)=CO(I,J,K,L)-CO(I-1,J-1,K,L)
 310    CO2(L)=CO(I,J-1,K,L)-CO(I-1,J,K,L)
        CALL AXB(CO1,CO2,CO3)
        AREA(I,J,K,3)=0.5*SQRT(CO3(1)**2+CO3(2)**2+CO3(3)**2)
        DO 320 L=1,3
 320    SZET(I,J,K,L)=0.5*CO3(L)
  300   CONTINUE
        DO 400 K=2,KN
        DO 400 J=2,JN
        DO 400 I=2,IN
        D1=0.0
        D2=0.0
        D3=0.0
        DO L=1,3
        CO1(L)=CO(I,J,K,L)-CO(I-1,J-1,K-1,L)
        D1=D1+SXI(I-1,J,K,L)*CO1(L)
        D2=D2+SETA(I,J-1,K,L)*CO1(L)
        D3=D3+SZET(I,J,K-1,L)*CO1(L)
        END DO
        VOL(I,J,K)=(ABS(D1)+ABS(D2)+ABS(D3))/3.0
 400    CONTINUE
C
C --- Normal Height Calculation --------
C
        DO 10 K=2,KN
        DO 10 I=2,IN
        X0=0.25*(CO(I-1,1,K,1)+CO(I,1,K,1)+CO(I,1,K-1,1)+CO(I-1,1,K-1,1))
        Y0=0.25*(CO(I-1,1,K,2)+CO(I,1,K,2)+CO(I,1,K-1,2)+CO(I-1,1,K-1,2))
        Z0=0.25*(CO(I-1,1,K,3)+CO(I,1,K,3)+CO(I,1,K-1,3)+CO(I-1,1,K-1,3))
        DS(I,1,K)=0.0
```

```
      DO 10 J=2,JN
      XX=.125*(CO(I-1,J,K,1)+CO(I,J,K,1)+CO(I,J,K-1,1)+CO(I-1,J,K-1,1)
     &+CO(I-1,J-1,K,1)+CO(I,J-1,K,1)+CO(I,J-1,K-1,1)+CO(I-1,J-1,K-1,1))
      YY=.125*(CO(I-1,J,K,2)+CO(I,J,K,2)+CO(I,J,K-1,2)+CO(I-1,J,K-1,2)
     &+CO(I-1,J-1,K,2)+CO(I,J-1,K,2)+CO(I,J-1,K-1,2)+CO(I-1,J-1,K-1,2))
      ZZ=.125*(CO(I-1,J,K,3)+CO(I,J,K,3)+CO(I,J,K-1,3)+CO(I-1,J,K-1,3)
     &+CO(I-1,J-1,K,3)+CO(I,J-1,K,3)+CO(I,J-1,K-1,3)+CO(I-1,J-1,K-1,3))
C
      SA=SQRT(SETA(I,1,K,1)**2+SETA(I,1,K,2)**2+SETA(I,1,K,3)**2)
      DS(I,J,K)=ABS((XX-X0)*SETA(I,1,K,1)+(YY-Y0)*SETA(I,1,K,2)+(ZZ-Z0)
     & *SETA(I,1,K,3))/SA
10    CONTINUE
C
      DO 510 J=2,JN
      DO 510 I=2,IN
      SXI(I,J,1,1)=-SXI(I,J,2,1)
      SXI(I,J,1,2)= SXI(I,J,2,2)
      SXI(I,J,1,3)= SXI(I,J,2,3)
      SETA(I,J,1,1)=-SETA(I,J,2,1)
      SETA(I,J,1,2)= SETA(I,J,2,2)
      SETA(I,J,1,3)= SETA(I,J,2,3)
      SXI(I,J,KN+1,1)=-SXI(I,J,KN,1)
      SXI(I,J,KN+1,2)= SXI(I,J,KN,2)
      SXI(I,J,KN+1,3)= SXI(I,J,KN,3)
      SETA(I,J,KN+1,1)=-SETA(I,J,KN,1)
      SETA(I,J,KN+1,2)= SETA(I,J,KN,2)
      SETA(I,J,KN+1,3)= SETA(I,J,KN,3)
      SZET(I,J,KN+1,1)= SZET(I,J,KN,1)
      SZET(I,J,KN+1,2)=-SZET(I,J,KN,2)
      SZET(I,J,KN+1,3)=-SZET(I,J,KN,3)
      VOL(I,J,1)=VOL(I,J,2)
      VOL(I,J,KN+1)=VOL(I,J,KN)
 510  CONTINUE
      RETURN
      END
C -------------------------------------------------------------------------------
      SUBROUTINE AXB(A,B,C,D)
      DIMENSION A(3),B(3),C(3)
      C(1)=A(2)*B(3)-A(3)*B(2)
      C(2)=A(3)*B(1)-A(1)*B(3)
      C(3)=A(1)*B(2)-A(2)*B(1)
      D=0.5*SQRT(C(1)**2+C(2)**2+C(3)**2)
      RETURN
      END
```

4.5 Subroutine INI3D

This subroutine initializes the flow field. If IRST is given as zero in the input file, the flow field is assigned to have the free stream conditions, if IRST=1 is for restarting case, that is, the flow field is read from five files storing the primitive variables (ρ,u,v,w,p), respectively, and carry on the calculation. The temperature, sound speed and viscosity fields are then generated from those primitive variables. All the result files for the primitive variables, temperature and Mach number are opened here, they will not be closed until the whole calculations finish.

```
      SUBROUTINE INI3D
      INCLUDE "comm.f"
      COMMON /STATE/ RHO(IM,JM,KM),U(IM,JM,KM),V(IM,JM,KM),W(IM,JM,KM)
```

```
     &          ,T(IM,JM,KM),XMU(IM,JM,KM)
C
C --- BOUNDARY CONDITIONS AT INFINITE
C
      IF(IRST.EQ.0)THEN
      DO 10 K=1,KN+1
      DO 10 J=2,JN+1
      DO 10 I=1,IN+1
      RHO(I,J,K)=1.0
      U(I,J,K)=0.0
      V(I,J,K)=SIN(ALPHA)
      W(I,J,K)=COS(ALPHA)
      T(I,J,K)=1.0
      XMU(I,J,K)=1.0
      XMUT(I,J,K)=0.0
 10   CONTINUE
C
C --- WALL BOUNDARY
C
      DO 20 K=1,KN+1
      DO 20 I=1,IN+1
      U(I,1,K)=0.0
      V(I,1,K)=0.0
      W(I,1,K)=0.0
      T(I,1,K)=TW/TINF
      RHO(I,1,K)=RHO(I,2,K)*T(I,2,K)/T(I,1,K)
      IF(T(I,1,K).LE.RTINFA) XMU(I,1,K)=T(I,1,K)
      IF(T(I,1,K).GT.RTINFA) XMU(I,1,K)=
     &T(I,1,K)*230.0*SQRT(T(I,1,K)*TINFA)/(T(I,1,K)*TINF+110.0)
 20   CONTINUE
C
      ELSE
C
      OPEN(1,FILE='rho.bin',FORM='UNFORMATTED')
      READ(1) (((RHO(I,J,K),I=1,IN+1),J=1,JN+1),K=1,KN+1)
      OPEN(2,FILE='u.bin',FORM='UNFORMATTED')
      READ(2) (((U(I,J,K),I=1,IN+1),J=1,JN+1),K=1,KN+1)
      OPEN(3,FILE='v.bin',FORM='UNFORMATTED')
      READ(3) (((V(I,J,K),I=1,IN+1),J=1,JN+1),K=1,KN+1)
      OPEN(4,FILE='w.bin',FORM='UNFORMATTED')
      READ(4) (((W(I,J,K),I=1,IN+1),J=1,JN+1),K=1,KN+1)
      OPEN(11,FILE='t.bin',FORM='UNFORMATTED')
      READ(11) (((T(I,J,K),I=1,IN+1),J=1,JN+1),K=1,KN+1)
      CLOSE(1)
      CLOSE(11)
      CLOSE(2)
      CLOSE(3)
      CLOSE(4)
      DO 30 K=1,KN+1
      DO 30 J=1,JN+1
      DO 30 I=1,IN+1
      IF(T(I,J,K).LE.RTINFA) XMU(I,J,K)=T(I,J,K)
      IF(T(I,J,K).GT.RTINFA) XMU(I,J,K)=
     & T(I,J,K)*230.0*SQRT(T(I,J,K)*TINFA)/(T(I,J,K)*TINF+110.0)
 30   CONTINUE
      ENDIF
      RETURN
      END
```

4.6 Subroutine BFL3D

This subroutine calculates both the inviscid and viscous fluxes at the boundaries of the given domain of a problem. Since FORTRAN compiler allocates arrays of a common block in a computer storage in such a way that they are actually connected consecutively together. In order to increase the efficiency of the calculation for both this module and ROSPNS, which are the two major modules being called for the calculation, the primitive variables (ρ, u, v, w, T) are put in such order that the EQUIVALENCE technique of FORTRAN can be used. PVR(IM,JM,KM,5) is the array for connecting the primitive variables in a sense of the column hyper-vector, which does not take any more space but equals to the allocations of the arrays for ρ, u, v, w, T. The array PVR could have been used instead of the other five arrays (even more) for the primitive variables for the rest of the program, it is only because of the readability of the code that PVR array is used as such here. The formulas for the fluxes evaluations refer to section 3.6.

```
      SUBROUTINE BFL3D
C
C To set up flow properties at boundaries and to calculate the inviscid
C and viscous fluxes at boundaries.
C
      INCLUDE "comm.f"
      COMMON /STATE/ RHO(IM,JM,KM),U(IM,JM,KM),V(IM,JM,KM),W(IM,JM,KM)
     &        ,T(IM,JM,KM),XMU(IM,JM,KM)
      DIMENSION PVR(IM,JM,KM,5)
      DIMENSION FIB(5),E(5),F(5),G(5),QL(5),QR(5)
      DIMENSION SXII(3,2),SETAI(3,2),SZETI(3,2)
      DIMENSION SXIJ(3,2),SETAJ(3,2),SZETJ(3,2)
      DIMENSION SXIK(3,2),SETAK(3,2),SZETK(3,2)
      EQUIVALENCE (RHO(1,1,1),PVR(1,1,1,1))
C
C --- downstream boundary interface flux EWAVE at (IN+1/2,j,k)
C
C Q(IN+1,J,K) represents values at (IN+1/2,J,K) and equals to QL
C
      I=IN
      I1=IN+1
      DO 200 K=2,KN
      DO 200 J=2,JN
C
      RHO(I1,J,K)=RHO(I,J,K)
      U(I1,J,K)=U(I,J,K)
      V(I1,J,K)=V(I,J,K)
      W(I1,J,K)=W(I,J,K)
      T(I1,J,K)=T(I,J,K)
      XMUT(I1,J,K)=XMUT(I,J,K)
      IF(T(I1,J,K).LE.RTINFA)THEN
      XMU(I1,J,K)=T(I1,J,K)
      ELSE
      XMU(I1,J,K)=T(I1,J,K)*230.0*SQRT(T(I1,J,K)*TINFA)
     & /(T(I1,J,K)*TINF+110.0)
      ENDIF
 200  CONTINUE
C
C Streamwise fluxes at (IN+1/2,J,K)
C
      DO 210 K=2,KN
```

```
      DO 210 J=2,JN
      VS02=0.5*(U(I1,J,K)**2+V(I1,J,K)**2+W(I1,J,K)**2)
      PP=T(I1,J,K)*RHO(I1,J,K)*GAMC6
      ET=PP*GAMC2+RHO(I1,J,K)*VS02
      UC=SXI(I,J,K,1)*U(I1,J,K)+SXI(I,J,K,2)*V(I1,J,K)
     &  +SXI(I,J,K,3)*W(I1,J,K)
      RR(I,J,K,1)=RHO(I1,J,K)*UC
      RR(I,J,K,2)=RR(I,J,K,1)*U(I1,J,K)+SXI(I,J,K,1)*PP
      RR(I,J,K,3)=RR(I,J,K,1)*V(I1,J,K)+SXI(I,J,K,2)*PP
      RR(I,J,K,4)=RR(I,J,K,1)*W(I1,J,K)+SXI(I,J,K,3)*PP
      RR(I,J,K,5)=ET*UC
210   CONTINUE
      DO 220 K=2,KN
      DO 220 J=2,JN
      C1=(XMU(I1,J,K)+XMUT(I1,J,K))*RRE
      C2=2.0*C1/3.0
      C3=2.0*(XMU(I1,J,K)*C0+XMUT(I1,J,K)*CT)
C
C Cell surface vector averages ------
C
      RECVOLM=2.0/(3.0*VOL(I,J,K)-VOL(I-1,J,K))
      SXII(1,1)=0.5*(SXI(I,J,K,1)+SXI(I-1,J,K,1))
      SXII(1,2)=0.5*(3.0*SXI(I,J,K,1)-SXI(I-1,J,K,1))
      SETAI(1,1)=0.5*(3.0*SETA(I,J-1,K,1)-SETA(I-1,J-1,K,1))
      SETAI(1,2)=0.5*(3.0*SETA(I,J,K,1)-SETA(I-1,J,K,1))
      SZETI(1,1)=0.5*(3.0*SZET(I,J,K-1,1)-SZET(I-1,J,K-1,1))
      SZETI(1,2)=0.5*(3.0*SZET(I,J,K,1)-SZET(I-1,J,K,1))
      SXII(2,1)=0.5*(SXI(I,J,K,2)+SXI(I-1,J,K,2))
      SXII(2,2)=0.5*(3.0*SXI(I,J,K,2)-SXI(I-1,J,K,2))
      SETAI(2,1)=0.5*(3.0*SETA(I,J-1,K,2)-SETA(I-1,J-1,K,2))
      SETAI(2,2)=0.5*(3.0*SETA(I,J,K,2)-SETA(I-1,J,K,2))
      SZETI(2,1)=0.5*(3.0*SZET(I,J,K-1,2)-SZET(I-1,J,K-1,2))
      SZETI(2,2)=0.5*(3.0*SZET(I,J,K,2)-SZET(I-1,J,K,2))
      SXII(3,1)=0.5*(SXI(I,J,K,3)+SXI(I-1,J,K,3))
      SXII(3,2)=0.5*(3.0*SXI(I,J,K,3)-SXI(I-1,J,K,3))
      SETAI(3,1)=0.5*(3.0*SETA(I,J-1,K,3)-SETA(I-1,J-1,K,3))
      SETAI(3,2)=0.5*(3.0*SETA(I,J,K,3)-SETA(I-1,J,K,3))
      SZETI(3,1)=0.5*(3.0*SZET(I,J,K-1,3)-SZET(I-1,J,K-1,3))
      SZETI(3,2)=0.5*(3.0*SZET(I,J,K,3)-SZET(I-1,J,K,3))
      UM=U(I1,J,K)
      VM=V(I1,J,K)
      WM=W(I1,J,K)
      UM11=0.5*(U(I1,J-1,K)+U(I1,J,K))
      UM12=0.5*(U(I1,J+1,K)+U(I1,J,K))
      UM21=0.5*(U(I1,J,K-1)+U(I1,J,K))
      UM22=0.5*(U(I1,J,K+1)+U(I1,J,K))
      VM11=0.5*(V(I1,J-1,K)+V(I1,J,K))
      VM12=0.5*(V(I1,J+1,K)+V(I1,J,K))
      VM21=0.5*(V(I1,J,K-1)+V(I1,J,K))
      VM22=0.5*(V(I1,J,K+1)+V(I1,J,K))
      WM11=0.5*(W(I1,J-1,K)+W(I1,J,K))
      WM12=0.5*(W(I1,J+1,K)+W(I1,J,K))
      WM21=0.5*(W(I1,J,K-1)+W(I1,J,K))
      WM22=0.5*(W(I1,J,K+1)+W(I1,J,K))
      TM11=0.5*(T(I1,J-1,K)+T(I1,J,K))
      TM12=0.5*(T(I1,J+1,K)+T(I1,J,K))
      TM21=0.5*(T(I1,J,K-1)+T(I1,J,K))
      TM22=0.5*(T(I1,J,K+1)+T(I1,J,K))
```

```
      PUPX=(SXII (1,2)*(2.0*U(I,J,K)-U(I-1,J,K))-SXII (1,1)*U(I,J,K)
     &    +SETAI(1,2)*UM12-SETAI(1,1)*UM11
     &    +SZETI(1,2)*UM22-SZETI(1,1)*UM21)*RECVOLM
      PUPY=(SXII (2,2)*(2.0*U(I,J,K)-U(I-1,J,K))-SXII (2,1)*U(I,J,K)
     &    +SETAI(2,2)*UM12-SETAI(2,1)*UM11
     &    +SZETI(2,2)*UM22-SZETI(2,1)*UM21)*RECVOLM
      PUPZ=(SXII (3,2)*(2.0*U(I,J,K)-U(I-1,J,K))-SXII (3,1)*U(I,J,K)
     &    +SETAI(3,2)*UM12-SETAI(3,1)*UM11
     &    +SZETI(3,2)*UM22-SZETI(3,1)*UM21)*RECVOLM
      PVPX=(SXII (1,2)*(2.0*V(I,J,K)-V(I-1,J,K))-SXII (1,1)*V(I,J,K)
     &    +SETAI(1,2)*VM12-SETAI(1,1)*VM11
     &    +SZETI(1,2)*VM22-SZETI(1,1)*VM21)*RECVOLM
      PVPY=(SXII (2,2)*(2.0*V(I,J,K)-V(I-1,J,K))-SXII (2,1)*V(I,J,K)
     &    +SETAI(2,2)*VM12-SETAI(2,1)*VM11
     &    +SZETI(2,2)*VM22-SZETI(2,1)*VM21)*RECVOLM
      PVPZ=(SXII (3,2)*(2.0*V(I,J,K)-V(I-1,J,K))-SXII (3,1)*V(I,J,K)
     &    +SETAI(3,2)*VM12-SETAI(3,1)*VM11
     &    +SZETI(3,2)*VM22-SZETI(3,1)*VM21)*RECVOLM
      PWPX=(SXII (1,2)*(2.0*W(I,J,K)-W(I-1,J,K))-SXII (1,1)*W(I,J,K)
     &    +SETAI(1,2)*WM12-SETAI(1,1)*WM11
     &    +SZETI(1,2)*WM22-SZETI(1,1)*WM21)*RECVOLM
      PWPY=(SXII (2,2)*(2.0*W(I,J,K)-W(I-1,J,K))-SXII (2,1)*W(I,J,K)
     &    +SETAI(2,2)*WM12-SETAI(2,1)*WM11
     &    +SZETI(2,2)*WM22-SZETI(2,1)*WM21)*RECVOLM
      PWPZ=(SXII (3,2)*(2.0*W(I,J,K)-W(I-1,J,K))-SXII (3,1)*W(I,J,K)
     &    +SETAI(3,2)*WM12-SETAI(3,1)*WM11
     &    +SZETI(3,2)*TM22-SZETI(3,1)*TM21)*RECVOLM
      PTPX=(SXII (1,2)*(2.0*T(I,J,K)-T(I-1,J,K))-SXII (1,1)*T(I,J,K)
     &    +SETAI(1,2)*TM12-SETAI(1,1)*TM11
     &    +SZETI(1,2)*TM22-SZETI(1,1)*TM21)*RECVOLM
      PTPY=(SXII (2,2)*(2.0*T(I,J,K)-T(I-1,J,K))-SXII (2,1)*T(I,J,K)
     &    +SETAI(2,2)*TM12-SETAI(2,1)*TM11
     &    +SZETI(2,2)*TM22-SZETI(2,1)*TM21)*RECVOLM
      PTPZ=(SXII (3,2)*(2.0*T(I,J,K)-T(I-1,J,K))-SXII (3,1)*T(I,J,K)
     &    +SETAI(3,2)*TM12-SETAI(3,1)*TM11
     &    +SZETI(3,2)*TM22-SZETI(3,1)*TM21)*RECVOLM
      TAUXX=C2*(2.0*PUPX-PVPY-PWPZ)
      TAUYY=C2*(2.0*PVPY-PUPX-PWPZ)
      TAUZZ=C2*(2.0*PWPZ-PUPX-PVPY)
      TAUXY=C1*(PUPY+PVPX)
      TAUXZ=C1*(PUPZ+PWPX)
      TAUYZ=C1*(PVPZ+PWPY)
      QX=-C3*PTPX
      QY=-C3*PTPY
      QZ=-C3*PTPZ
      E(2)=TAUXX
      E(3)=TAUXY
      E(4)=TAUXZ
      E(5)=U(I1,J,K)*TAUXX+V(I1,J,K)*TAUXY+W(I1,J,K)*TAUXZ-QX
      F(2)=TAUXY
      F(3)=TAUYY
      F(4)=TAUYZ
      F(5)=U(I1,J,K)*TAUXY+V(I1,J,K)*TAUYY+W(I1,J,K)*TAUYZ-QY
      G(2)=TAUXZ
      G(3)=TAUYZ
      G(4)=TAUZZ
      G(5)=U(I1,J,K)*TAUXZ+V(I1,J,K)*TAUYZ+W(I1,J,K)*TAUZZ-QZ
      RR(I,J,K,2)=RR(I,J,K,2)-(SXI(I,J,K,1)*E(2)+SXI(I,J,K,2)*F(2)
```

```
     &              +SXI(I,J,K,3)*G(2))
       RR(I,J,K,3)=RR(I,J,K,3)-(SXI(I,J,K,1)*E(3)+SXI(I,J,K,2)*F(3)
     &              +SXI(I,J,K,3)*G(3))
       RR(I,J,K,4)=RR(I,J,K,4)-(SXI(I,J,K,1)*E(4)+SXI(I,J,K,2)*F(4)
     &              +SXI(I,J,K,3)*G(4))
       RR(I,J,K,5)=RR(I,J,K,5)-(SXI(I,J,K,1)*E(5)+SXI(I,J,K,2)*F(5)
     &              +SXI(I,J,K,3)*G(5))
 220   CONTINUE
C
C --- wall boundary interface flux FWAVE at (i,1+1/2,k)
C
       J=1
       DO 300 K=2,KN
       DO 300 I=2,IN
C
       DO 40 L=1,5
       DM=2.0*(PVR(I,2,K,L)-PVR(I,1,K,L))
       DP=PVR(I,3,K,L)-PVR(I,2,K,L)
       S=(2.0*DP*DM+EPS)/(DP*DP+DM*DM+EPS)
       QR(L)=PVR(I,2,K,L)-0.25*S*((1.0+CK*S)*DM+(1.0-CK*S)*DP)
 40    CONTINUE
C
       RHOR=QR(1)
       PRR=QR(5)*RHOR*GAMC6
       CSR=SQRT(GAM*QR(5)*GAMC6)
       UBR=SETA(I,1,K,1)*QR(2)+SETA(I,1,K,2)*QR(3)+SETA(I,1,K,3)*QR(4)
C
C boundary interface Riemann solver
C
       CSB=CSR-0.5*(GAM-1.0)*UBR/AREA(I,1,K,2)
       RHOB=(CSB*CSB*GAMC3*RHOR**GAM/PRR)**GAMC4
       PB=RHOB*CSB**2*GAMC3
       RR(I,2,K,2)=RR(I,2,K,2)-SETA(I,1,K,1)*PB
       RR(I,2,K,3)=RR(I,2,K,3)-SETA(I,1,K,2)*PB
       RR(I,2,K,4)=RR(I,2,K,4)-SETA(I,1,K,3)*PB
C
C The inviscid and viscous boundary conditions are separated
C
       RHO(I,1,K)=RHO(I,2,K)*T(I,2,K)/T(I,1,K)
 300   CONTINUE
C
       J=1
       DO 320 K=2,KN
       DO 320 I=2,IN
       C1=(XMU(I,1,K)+XMUT(I,1,K))*RRE
       C2=2.0*C1/3.0
       C3=2.0*(XMU(I,1,K)*C0+XMUT(I,1,K)*CT)
       RECVOLM=2.0/(3.0*VOL(I,2,K)-VOL(I,3,K))
       SXIJ (1,1)=0.5*(3.0*SXI(I-1,2,K,1)-SXI(I-1,3,K,1))
       SXIJ (1,2)=0.5*(3.0*SXI(I,2,K,1)-SXI(I,3,K,1))
       SETAJ(1,1)=0.5*(3.0*SETA(I,1,K,1)-SETA(I,2,K,1))
C
       SETAJ(1,2)=0.5*(SETA(I,2,K,1)+SETA(I,1,K,1))
       SZETJ(1,1)=0.5*(3.0*SZET(I,2,K-1,1)-SZET(I,3,K-1,1))
       SZETJ(1,2)=0.5*(3.0*SZET(I,2,K,1)-SZET(I,3,K,1))
       SXIJ (2,1)=0.5*(3.0*SXI(I-1,2,K,2)-SXI(I-1,3,K,2))
       SXIJ (2,2)=0.5*(3.0*SXI(I,2,K,2)-SXI(I,3,K,2))
       SETAJ(2,1)=0.5*(3.0*SETA(I,1,K,2)-SETA(I,2,K,2))
```

```
C
      SETAJ(2,2)=0.5*(SETA(I,2,K,2)+SETA(I,1,K,2))
      SZETJ(2,1)=0.5*(3.0*SZET(I,2,K-1,2)-SZET(I,3,K-1,2))
      SZETJ(2,2)=0.5*(3.0*SZET(I,2,K,2)-SZET(I,3,K,2))
      SXIJ (3,1)=0.5*(3.0*SXI(I-1,2,K,3)-SXI(I-1,3,K,3))
      SXIJ (3,2)=0.5*(3.0*SXI(I,2,K,3)-SXI(I,3,K,3))
      SETAJ(3,1)=0.5*(3.0*SETA(I,1,K,3)-SETA(I,2,K,3))
C
      SETAJ(3,2)=0.5*(SETA(I,2,K,3)+SETA(I,1,K,3))
      SZETJ(3,1)=0.5*(3.0*SZET(I,2,K-1,3)-SZET(I,3,K-1,3))
      SZETJ(3,2)=0.5*(3.0*SZET(I,2,K,3)-SZET(I,3,K,3))
      PUPX=(SETAJ(1,2)+SETAJ(1,1))*U(I,2,K)*RECVOLM
      PUPY=(SETAJ(2,2)+SETAJ(2,1))*U(I,2,K)*RECVOLM
      PUPZ=(SETAJ(3,2)+SETAJ(3,1))*U(I,2,K)*RECVOLM
      PVPX=(SETAJ(1,2)+SETAJ(1,1))*V(I,2,K)*RECVOLM
      PVPY=(SETAJ(2,2)+SETAJ(2,1))*V(I,2,K)*RECVOLM
      PVPZ=(SETAJ(3,2)+SETAJ(3,1))*V(I,2,K)*RECVOLM
      PWPX=(SETAJ(1,2)+SETAJ(1,1))*W(I,2,K)*RECVOLM
      PWPY=(SETAJ(2,2)+SETAJ(2,1))*W(I,2,K)*RECVOLM
      PWPZ=(SETAJ(3,2)+SETAJ(3,1))*W(I,2,K)*RECVOLM
      PTPX=(SETAJ(1,2)*T(I,J+1,K)-SETAJ(1,1)*(2.0*T(I,1,K)-T(I,2,K))
     &    +(SXIJ (1,2)-SXIJ (1,1)
     &    + SZETJ(1,2)-SZETJ(1,1))*T(I,1,K))*RECVOLM
      PTPY=(SETAJ(2,2)*T(I,J+1,K)-SETAJ(2,1)*(2.0*T(I,1,K)-T(I,2,K))
     &    +(SXIJ (2,2)-SXIJ (2,1)
     &    + SZETJ(2,2)-SZETJ(2,1))*T(I,1,K))*RECVOLM
      PTPZ=(SETAJ(3,2)*T(I,J+1,K)-SETAJ(3,1)*(2.0*T(I,1,K)-T(I,2,K))
     &    +(SXIJ (3,2)-SXIJ (3,1)
     &    + SZETJ(3,2)-SZETJ(3,1))*T(I,1,K))*RECVOLM
      TAUXX=C2*(2.0*PUPX-PVPY-PWPZ)
      TAUYY=C2*(2.0*PVPY-PUPX-PWPZ)
      TAUZZ=C2*(2.0*PWPZ-PUPX-PVPY)
      TAUXY=C1*(PUPY+PVPX)
      TAUXZ=C1*(PUPZ+PWPX)
      TAUYZ=C1*(PVPZ+PWPY)
      QX=C3*PTPX
      QY=C3*PTPY
      QZ=C3*PTPZ
      E(2)=TAUXX
      E(3)=TAUXY
      E(4)=TAUXZ
      E(5)=QX
      F(2)=TAUXY
      F(3)=TAUYY
      F(4)=TAUYZ
      F(5)=QY
      G(2)=TAUXZ
      G(3)=TAUYZ
      G(4)=TAUZZ
      G(5)=QZ
      RR(I,2,K,2)=RR(I,2,K,2)+(SETA(I,1,K,1)*E(2)+SETA(I,1,K,2)*F(2)
     &           +SETA(I,1,K,3)*G(2))
      RR(I,2,K,3)=RR(I,2,K,3)+(SETA(I,1,K,1)*E(3)+SETA(I,1,K,2)*F(3)
     &           +SETA(I,1,K,3)*G(3))
      RR(I,2,K,4)=RR(I,2,K,4)+(SETA(I,1,K,1)*E(4)+SETA(I,1,K,2)*F(4)
     &           +SETA(I,1,K,3)*G(4))
      RR(I,2,K,5)=RR(I,2,K,5)+(SETA(I,1,K,1)*E(5)+SETA(I,1,K,2)*F(5)
     &           +SETA(I,1,K,3)*G(5))
```

```
 320     CONTINUE
C
C --- outer boundary interface flux FWAVE at (i,JN+1,k)
C
         J1=JN+1
         DO 410 K=2,KN
         DO 410 I=2,IN
         VS02=0.5*(U(I,J1,K)**2+V(I,J1,K)**2+W(I,J1,K)**2)
         PP=T(I,J1,K)*RHO(I,J1,K)*GAMC6
         ET=PP*GAMC2+RHO(I,J1,K)*VS02
         UC=SETA(I,JN,K,1)*U(I,J1,K)+SETA(I,JN,K,2)*V(I,J1,K)
        &  +SETA(I,JN,K,3)*W(I,J1,K)
         FN1=RHO(I,J1,K)*UC
         RR(I,JN,K,1)=RR(I,JN,K,1)+FN1
         RR(I,JN,K,2)=RR(I,JN,K,2)+FN1*U(I,J1,K)+SETA(I,JN,K,1)*PP
         RR(I,JN,K,3)=RR(I,JN,K,3)+FN1*V(I,J1,K)+SETA(I,JN,K,2)*PP
         RR(I,JN,K,4)=RR(I,JN,K,4)+FN1*W(I,J1,K)+SETA(I,JN,K,3)*PP
         RR(I,JN,K,5)=RR(I,JN,K,5)+ET*UC
 410     CONTINUE
C
C --- symmetric boundary fluxes GWAVE at (i,j,1+1/2) (i,j,KN+1/2)
C
         K1=KN+1
         DO 500 J=2,JN
         DO 500 I=2,IN
         XMUT(I,J,1)=XMUT(I,J,2)
         RHO(I,J,1)=RHO(I,J,2)
         U(I,J,1)=-U(I,J,2)
         V(I,J,1)=V(I,J,2)
         W(I,J,1)=W(I,J,2)
         T(I,J,1)=T(I,J,2)
         IF(T(I,J,1).LE.RTINFA)XMU(I,J,1)=T(I,J,1)
         IF(T(I,J,1).GT.RTINFA)XMU(I,J,1)=
        &T(I,J,1)*230.0*SQRT(T(I,J,1)*TINFA)/(T(I,J,1)*TINF+110.0)
         XMUT(I,J,K1)=XMUT(I,J,KN)
         RHO(I,J,K1)=RHO(I,J,KN)
         U(I,J,K1)=-U(I,J,KN)
         V(I,J,K1)=V(I,J,KN)
         W(I,J,K1)=W(I,J,KN)
         T(I,J,K1)=T(I,J,KN)
         IF(T(I,J,K1).LE.RTINFA)XMU(I,J,K1)=T(I,J,K1)
         IF(T(I,J,K1).GT.RTINFA)XMU(I,J,K1)=T(I,J,K1)*230.0*
        &SQRT(T(I,J,K1)*TINFA)/(T(I,J,K1)*TINF+110.0)
 500     CONTINUE
C
         DO 5000 J=2,JN
         DO 5000 I=2,IN
         DM=RHO(I,J,2)-RHO(I,J,3)
         S=EPS/(DM*DM+EPS)
         RH1L=RHO(I,J,1)+0.25*S*(1.0-CK*S)*DM
         DM=T(I,J,2)-T(I,J,3)
         S=EPS/(DM*DM+EPS)
         T1L=T(I,J,1)+0.25*S*(1.0-CK*S)*DM
         DM=RHO(I,J,KN)-RHO(I,J,KN-1)
         S=EPS/(DM*DM+EPS)
         RHNL=RHO(I,J,KN)+0.25*S*(1.0-CK*S)*DM
         DM=T(I,J,KN)-T(I,J,KN-1)
         S=EPS/(DM*DM+EPS)
```

```
          TNL=T(I,J,KN)+0.25*S*(1.0-CK*S)*DM
C
C---- VELOCITY COMPONENTS IN THE DIRECTION NORMAL TO THE SYMMETRICAL
C     SURFACE IS ZERO (UH=0.).
C
          P1L=T1L*RH1L*GAMC6
          RR(I,J,2,2)=RR(I,J,2,2)-P1L*SZET(I,J,1,1)
          RR(I,J,2,3)=RR(I,J,2,3)-P1L*SZET(I,J,1,2)
          RR(I,J,2,4)=RR(I,J,2,4)-P1L*SZET(I,J,1,3)
C
          PNL=TNL*RHNL*GAMC6
          RR(I,J,KN,2)=RR(I,J,KN,2)+PNL*SZET(I,J,KN,1)
          RR(I,J,KN,3)=RR(I,J,KN,3)+PNL*SZET(I,J,KN,2)
          RR(I,J,KN,4)=RR(I,J,KN,4)+PNL*SZET(I,J,KN,3)
C
 5000     CONTINUE
C
          K=1
          DO 520 J=2,JN
          DO 520 I=2,IN
C
          C1=0.5*(XMU(I,J,K)+XMU(I,J,K+1)+XMUT(I,J,K)+XMUT(I,J,K+1))*RRE
          C2=2.0*C1/3.0
          C3=(XMU(I,J,K)+XMU(I,J,K+1))*C0+(XMUT(I,J,K)+XMUT(I,J,K+1))*CT
          SXIK(1,1)=0.5*(SXI(I-1,J,K,1)+SXI(I-1,J,K+1,1))
          SXIK(1,2)=0.5*(SXI(I,J,K,1)+SXI(I,J,K+1,1))
          SETAK(1,1)=0.5*(SETA(I,J-1,K,1)+SETA(I,J-1,K+1,1))
          SETAK(1,2)=0.5*(SETA(I,J,K,1)+SETA(I,J,K+1,1))
          SZETK(1,1)=0.5*(SZET(I,J,K,1)+SZET(I,J,K+1,1))
C -- ATTENTION!! SZET(I,J,0,1)=SZET(I,J,2,1)
          SZETK(1,2)=0.5*(SZET(I,J,K,1)+SZET(I,J,K+1,1))
          SXIK(2,1)=0.5*(SXI(I-1,J,K,2)+SXI(I-1,J,K+1,2))
          SXIK(2,2)=0.5*(SXI(I,J,K,2)+SXI(I,J,K+1,2))
          SETAK(2,1)=0.5*(SETA(I,J-1,K,2)+SETA(I,J-1,K+1,2))
          SETAK(2,2)=0.5*(SETA(I,J,K,2)+SETA(I,J,K+1,2))
          SZETK(2,1)=0.5*(SZET(I,J,K,2)-SZET(I,J,K+1,2))
C -- ATTENTION!! SZET(I,J,0,2)=-SZET(I,J,2,2)
          SZETK(2,2)=0.5*(SZET(I,J,K,2)+SZET(I,J,K+1,2))
          SXIK(3,1)=0.5*(SXI(I-1,J,K,3)+SXI(I-1,J,K+1,3))
          SXIK(3,2)=0.5*(SXI(I,J,K,3)+SXI(I,J,K+1,3))
          SETAK(3,1)=0.5*(SETA(I,J-1,K,3)+SETA(I,J-1,K+1,3))
          SETAK(3,2)=0.5*(SETA(I,J,K,3)+SETA(I,J,K+1,3))
C -- ATTENTION!! SZET(I,J,0,3)=-SZET(I,J,2,3)
          SZETK(3,1)=0.5*(SZET(I,J,K,3)-SZET(I,J,K+1,3))
          SZETK(3,2)=0.5*(SZET(I,J,K,3)+SZET(I,J,K+1,3))
          UM=0.5*(U(I,J,K+1)+U(I,J,K))
          VM=0.5*(V(I,J,K+1)+V(I,J,K))
          WM=0.5*(W(I,J,K+1)+W(I,J,K))
          RECVOLM=2.0/(VOL(I,J,K)+VOL(I,J,K+1))
          UM11=0.0
          UM12=0.0
          UM21=0.0
          UM22=0.0
          VM11=0.25*(V(I-1,J,K+1)+V(I,J,K+1)+V(I-1,J,K)+V(I,J,K))
          VM12=0.25*(V(I+1,J,K+1)+V(I,J,K+1)+V(I+1,J,K)+V(I,J,K))
          VM21=0.25*(V(I,J-1,K+1)+V(I,J-1,K)+V(I,J,K+1)+V(I,J,K))
          VM22=0.25*(V(I,J+1,K+1)+V(I,J+1,K)+V(I,J,K+1)+V(I,J,K))
          WM11=0.25*(W(I-1,J,K+1)+W(I,J,K+1)+W(I-1,J,K)+W(I,J,K))
```

```
      WM12=0.25*(W(I+1,J,K+1)+W(I,J,K+1)+W(I+1,J,K)+W(I,J,K))
      WM21=0.25*(W(I,J-1,K+1)+W(I,J-1,K)+W(I,J,K+1)+W(I,J,K))
      WM22=0.25*(W(I,J+1,K+1)+W(I,J+1,K)+W(I,J,K+1)+W(I,J,K))
      TM11=0.25*(T(I-1,J,K+1)+T(I,J,K+1)+T(I-1,J,K)+T(I,J,K))
      TM12=0.25*(T(I+1,J,K+1)+T(I,J,K+1)+T(I+1,J,K)+T(I,J,K))
      TM21=0.25*(T(I,J-1,K+1)+T(I,J-1,K)+T(I,J,K+1)+T(I,J,K))
      TM22=0.25*(T(I,J+1,K+1)+T(I,J+1,K)+T(I,J,K+1)+T(I,J,K))
      RECVOLM=2.0/(VOL(I,J,K)+VOL(I,J,K+1))
      PUPX=(SZETK(1,2)*U(I,J,K+1)-SZETK(1,1)*U(I,J,K)
     &    +SETAK(1,2)*UM22-SETAK(1,1)*UM21
     &    +SXIK (1,2)*UM12-SXIK (1,1)*UM11)*RECVOLM
      PUPY=(SZETK(2,2)*U(I,J,K+1)-SZETK(2,1)*U(I,J,K)
     &    +SETAK(2,2)*UM22-SETAK(2,1)*UM21
     &    +SXIK (2,2)*UM12-SXIK (2,1)*UM11)*RECVOLM
      PUPZ=(SZETK(3,2)*U(I,J,K+1)-SZETK(3,1)*U(I,J,K)
     &    +SETAK(3,2)*UM22-SETAK(3,1)*UM21
     &    +SXIK (3,2)*UM12-SXIK (3,1)*UM11)*RECVOLM
      PVPX=(SZETK(1,2)*V(I,J,K+1)-SZETK(1,1)*V(I,J,K)
     &    +SETAK(1,2)*VM22-SETAK(1,1)*VM21
     &    +SXIK (1,2)*VM12-SXIK (1,1)*VM11)*RECVOLM
      PVPY=(SZETK(2,2)*V(I,J,K+1)-SZETK(2,1)*V(I,J,K)
     &    +SETAK(2,2)*VM22-SETAK(2,1)*VM21
     &    +SXIK (2,2)*VM12-SXIK (2,1)*VM11)*RECVOLM
      PVPZ=(SZETK(3,2)*V(I,J,K+1)-SZETK(3,1)*V(I,J,K)
     &    +SETAK(3,2)*VM22-SETAK(3,1)*VM21
     &    +SXIK (3,2)*VM12-SXIK (3,1)*VM11)*RECVOLM
      PWPX=(SZETK(1,2)*W(I,J,K+1)-SZETK(1,1)*W(I,J,K)
     &    +SETAK(1,2)*WM22-SETAK(1,1)*WM21
     &    +SXIK (1,2)*WM12-SXIK (1,1)*WM11)*RECVOLM
      PWPY=(SZETK(2,2)*W(I,J,K+1)-SZETK(2,1)*W(I,J,K)
     &    +SETAK(2,2)*WM22-SETAK(2,1)*WM21
     &    +SXIK (2,2)*WM12-SXIK (2,1)*WM11)*RECVOLM
      PWPZ=(SZETK(3,2)*W(I,J,K+1)-SZETK(3,1)*W(I,J,K)
     &    +SETAK(3,2)*WM22-SETAK(3,1)*WM21
     &    +SXIK (3,2)*WM12-SXIK (3,1)*WM11)*RECVOLM
      PTPX=(SZETK(1,2)*T(I,J,K+1)-SZETK(1,1)*T(I,J,K)
     &    +SETAK(1,2)*TM22-SETAK(1,1)*TM21
     &    +SXIK (1,2)*TM12-SXIK (1,1)*TM11)*RECVOLM
      PTPY=(SZETK(2,2)*T(I,J,K+1)-SZETK(2,1)*T(I,J,K)
     &    +SETAK(2,2)*TM22-SETAK(2,1)*TM21
     &    +SXIK (2,2)*TM12-SXIK (2,1)*TM11)*RECVOLM
      PTPZ=(SZETK(3,2)*T(I,J,K+1)-SZETK(3,1)*T(I,J,K)
     &    +SETAK(3,2)*TM22-SETAK(3,1)*TM21
     &    +SXIK (3,2)*TM12-SXIK (3,1)*TM11)*RECVOLM
      TAUXX=C2*(2.0*PUPX-PVPY-PWPZ)
      TAUYY=C2*(2.0*PVPY-PUPX-PWPZ)
      TAUZZ=C2*(2.0*PWPZ-PUPX-PVPY)
      TAUXY=C1*(PUPY+PVPX)
      TAUXZ=C1*(PUPZ+PWPX)
      TAUYZ=C1*(PVPZ+PWPY)
      QX=-C3*PTPX
      QY=-C3*PTPY
      QZ=-C3*PTPZ
      E(2)=TAUXX
      E(3)=TAUXY
      E(4)=TAUXZ
      E(5)=UM*TAUXX+VM*TAUXY+WM*TAUXZ-QX
      F(2)=TAUXY
```

```
      F(3)=TAUYY
      F(4)=TAUYZ
      F(5)=UM*TAUXY+VM*TAUYY+WM*TAUYZ-QY
      G(2)=TAUXZ
      G(3)=TAUYZ
      G(4)=TAUZZ
      G(5)=UM*TAUXZ+VM*TAUYZ+WM*TAUZZ-QZ
      RR(I,J,2,2)=RR(I,J,2,2)+(SZET(I,J,K,1)*E(2)+SZET(I,J,K,2)*F(2)
     &           +SZET(I,J,K,3)*G(2))
      RR(I,J,2,3)=RR(I,J,2,3)+(SZET(I,J,K,1)*E(3)+SZET(I,J,K,2)*F(3)
     &           +SZET(I,J,K,3)*G(3))
      RR(I,J,2,4)=RR(I,J,2,4)+(SZET(I,J,K,1)*E(4)+SZET(I,J,K,2)*F(4)
     &           +SZET(I,J,K,3)*G(4))
      RR(I,J,2,5)=RR(I,J,2,5)+(SZET(I,J,K,1)*E(5)+SZET(I,J,K,2)*F(5)
     &           +SZET(I,J,K,3)*G(5))
  520 CONTINUE
C
      K=KN
      DO 530 J=2,JN
      DO 530 I=2,IN
C
      C1=0.5*(XMU(I,J,K)+XMU(I,J,K+1)+XMUT(I,J,K)+XMUT(I,J,K+1))*RRE
      C2=2.0*C1/3.0
      C3=(XMU(I,J,K)+XMU(I,J,K+1))*C0+(XMUT(I,J,K)+XMUT(I,J,K+1))*CT
      SXIK(1,1)=0.5*(SXI(I-1,J,K,1)+SXI(I-1,J,K+1,1))
      SXIK(1,2)=0.5*(SXI(I,J,K,1)+SXI(I,J,K+1,1))
      SETAK(1,1)=0.5*(SETA(I,J-1,K,1)+SETA(I,J-1,K+1,1))
      SETAK(1,2)=0.5*(SETA(I,J,K,1)+SETA(I,J,K+1,1))
      SZETK(1,1)=0.5*(SZET(I,J,K,1)+SZET(I,J,K-1,1))
      SZETK(1,2)=0.5*(SZET(I,J,K,1)+SZET(I,J,K+1,1))
      SXIK(2,1)=0.5*(SXI(I-1,J,K,2)+SXI(I-1,J,K+1,2))
      SXIK(2,2)=0.5*(SXI(I,J,K,2)+SXI(I,J,K+1,2))
      SETAK(2,1)=0.5*(SETA(I,J-1,K,2)+SETA(I,J-1,K+1,2))
      SETAK(2,2)=0.5*(SETA(I,J,K,2)+SETA(I,J,K+1,2))
      SZETK(2,1)=0.5*(SZET(I,J,K,2)+SZET(I,J,K-1,2))
      SZETK(2,2)=0.5*(SZET(I,J,K,2)+SZET(I,J,K+1,2))
      SXIK(3,1)=0.5*(SXI(I-1,J,K,3)+SXI(I-1,J,K+1,3))
      SXIK(3,2)=0.5*(SXI(I,J,K,3)+SXI(I,J,K+1,3))
      SETAK(3,1)=0.5*(SETA(I,J-1,K,3)+SETA(I,J-1,K+1,3))
      SETAK(3,2)=0.5*(SETA(I,J,K,3)+SETA(I,J,K+1,3))
      SZETK(3,1)=0.5*(SZET(I,J,K,3)+SZET(I,J,K-1,3))
      SZETK(3,2)=0.5*(SZET(I,J,K,3)+SZET(I,J,K+1,3))
      UM=U(I,J,K)
      VM=V(I,J,K)
      WM=W(I,J,K)
      RECVOLM=1.0/VOL(I,J,K)
      UM11=0.0
      UM12=0.0
      UM21=0.0
      UM22=0.0
      VM11=0.25*(V(I-1,J,K+1)+V(I,J,K+1)+V(I-1,J,K)+V(I,J,K))
      VM12=0.25*(V(I+1,J,K+1)+V(I,J,K+1)+V(I+1,J,K)+V(I,J,K))
      VM21=0.25*(V(I,J-1,K+1)+V(I,J-1,K)+V(I,J,K+1)+V(I,J,K))
      VM22=0.25*(V(I,J+1,K+1)+V(I,J+1,K)+V(I,J,K+1)+V(I,J,K))
      WM11=0.25*(W(I-1,J,K+1)+W(I,J,K+1)+W(I-1,J,K)+W(I,J,K))
      WM12=0.25*(W(I+1,J,K+1)+W(I,J,K+1)+W(I+1,J,K)+W(I,J,K))
      WM21=0.25*(W(I,J-1,K+1)+W(I,J-1,K)+W(I,J,K+1)+W(I,J,K))
      WM22=0.25*(W(I,J+1,K+1)+W(I,J+1,K)+W(I,J,K+1)+W(I,J,K))
```

```
      TM11=0.25*(T(I-1,J,K+1)+T(I,J,K+1)+T(I-1,J,K)+T(I,J,K))
      TM12=0.25*(T(I+1,J,K+1)+T(I,J,K+1)+T(I+1,J,K)+T(I,J,K))
      TM21=0.25*(T(I,J-1,K+1)+T(I,J-1,K)+T(I,J,K+1)+T(I,J,K))
      TM22=0.25*(T(I,J+1,K+1)+T(I,J+1,K)+T(I,J,K+1)+T(I,J,K))
      RECVOLM=2.0/(VOL(I,J,K)+VOL(I,J,K+1))
      PUPX=(SZETK(1,2)*U(I,J,K+1)-SZETK(1,1)*U(I,J,K)
     &    +SETAK(1,2)*UM22-SETAK(1,1)*UM21
     &    +SXIK (1,2)*UM12-SXIK (1,1)*UM11)*RECVOLM
      PUPY=(SZETK(2,2)*U(I,J,K+1)-SZETK(2,1)*U(I,J,K)
     &    +SETAK(2,2)*UM22-SETAK(2,1)*UM21
     &    +SXIK (2,2)*UM12-SXIK (2,1)*UM11)*RECVOLM
      PUPZ=(SZETK(3,2)*U(I,J,K+1)-SZETK(3,1)*U(I,J,K)
     &    +SETAK(3,2)*UM22-SETAK(3,1)*UM21
     &    +SXIK (3,2)*UM12-SXIK (3,1)*UM11)*RECVOLM
      PVPX=(SZETK(1,2)*V(I,J,K+1)-SZETK(1,1)*V(I,J,K)
     &    +SETAK(1,2)*VM22-SETAK(1,1)*VM21
     &    +SXIK (1,2)*VM12-SXIK (1,1)*VM11)*RECVOLM
      PVPY=(SZETK(2,2)*V(I,J,K+1)-SZETK(2,1)*V(I,J,K)
     &    +SETAK(2,2)*VM22-SETAK(2,1)*VM21
     &    +SXIK (2,2)*VM12-SXIK (2,1)*VM11)*RECVOLM
      PVPZ=(SZETK(3,2)*V(I,J,K+1)-SZETK(3,1)*V(I,J,K)
     &    +SETAK(3,2)*VM22-SETAK(3,1)*VM21
     &    +SXIK (3,2)*VM12-SXIK (3,1)*VM11)*RECVOLM
      PWPX=(SZETK(1,2)*W(I,J,K+1)-SZETK(1,1)*W(I,J,K)
     &    +SETAK(1,2)*WM22-SETAK(1,1)*WM21
     &    +SXIK (1,2)*WM12-SXIK (1,1)*WM11)*RECVOLM
      PWPY=(SZETK(2,2)*W(I,J,K+1)-SZETK(2,1)*W(I,J,K)
     &    +SETAK(2,2)*WM22-SETAK(2,1)*WM21
     &    +SXIK (2,2)*WM12-SXIK (2,1)*WM11)*RECVOLM
      PWPZ=(SZETK(3,2)*W(I,J,K+1)-SZETK(3,1)*W(I,J,K)
     &    +SETAK(3,2)*WM22-SETAK(3,1)*WM21
     &    +SXIK (3,2)*WM12-SXIK (3,1)*WM11)*RECVOLM
      PTPX=(SZETK(1,2)*T(I,J,K+1)-SZETK(1,1)*T(I,J,K)
     &    +SETAK(1,2)*TM22-SETAK(1,1)*TM21
     &    +SXIK (1,2)*TM12-SXIK (1,1)*TM11)*RECVOLM
      PTPY=(SZETK(2,2)*T(I,J,K+1)-SZETK(2,1)*T(I,J,K)
     &    +SETAK(2,2)*TM22-SETAK(2,1)*TM21
     &    +SXIK (2,2)*TM12-SXIK (2,1)*TM11)*RECVOLM
      PTPZ=(SZETK(3,2)*T(I,J,K+1)-SZETK(3,1)*T(I,J,K)
     &    +SETAK(3,2)*TM22-SETAK(3,1)*TM21
     &    +SXIK (3,2)*TM12-SXIK (3,1)*TM11)*RECVOLM
      TAUXX=C2*(2.0*PUPX-PVPY-PWPZ)
      TAUYY=C2*(2.0*PVPY-PUPX-PWPZ)
      TAUZZ=C2*(2.0*PWPZ-PUPX-PVPY)
      TAUXY=C1*(PUPY+PVPX)
      TAUXZ=C1*(PUPZ+PWPX)
      TAUYZ=C1*(PVPZ+PWPY)
      QX=-C3*PTPX
      QY=-C3*PTPY
      QZ=-C3*PTPZ
      E(2)=TAUXX
      E(3)=TAUXY
      E(4)=TAUXZ
      E(5)=UM*TAUXX+VM*TAUXY+WM*TAUXZ-QX
      F(2)=TAUXY
      F(3)=TAUYY
      F(4)=TAUYZ
      F(5)=UM*TAUXY+VM*TAUYY+WM*TAUYZ-QY
```

```
        G(2)=TAUXZ
        G(3)=TAUYZ
        G(4)=TAUZZ
        G(5)=UM*TAUXZ+VM*TAUYZ+WM*TAUZZ-QZ
        RR(I,J,K,2)=RR(I,J,K,2)-(SZET(I,J,K,1)*E(2)+SZET(I,J,K,2)*F(2)
       &              +SZET(I,J,K,3)*G(2))
        RR(I,J,K,3)=RR(I,J,K,3)-(SZET(I,J,K,1)*E(3)+SZET(I,J,K,2)*F(3)
       &              +SZET(I,J,K,3)*G(3))
        RR(I,J,K,4)=RR(I,J,K,4)-(SZET(I,J,K,1)*E(4)+SZET(I,J,K,2)*F(4)
       &              +SZET(I,J,K,3)*G(4))
        RR(I,J,K,5)=RR(I,J,K,5)-(SZET(I,J,K,1)*E(5)+SZET(I,J,K,2)*F(5)
       &              +SZET(I,J,K,3)*G(5))
  530   CONTINUE
C
C --- sharp nose boundary: free stream condition
C
        IF(ISHARP.EQ.1)THEN
        DO 10 K=1,KN+1
        DO 10 J=2,JN+1
        RHO(1,J,K)=1.0
        U(1,J,K)=0.0
        V(1,J,K)=SIN(ALPHA)
        W(1,J,K)=COS(ALPHA)
        T(1,J,K)=1.0
        XMU(1,J,K)=1.0
        XMUT(1,J,K)=0.0
   10   CONTINUE
        RETURN
        ENDIF
C
C --- polar singularity boundary interface flux EWAVE at (1+1/2,j,k)
C
        DO 100 K=2,KN
        DO 100 J=2,JN
        RHO(1,J,K)=0.5*(RHO(2,J,1)+RHO(2,J,KN+1))
        U(1,J,K)=0.0
        V(1,J,K)=0.5*(V(2,J,1)+V(2,J,KN+1))
        W(1,J,K)=0.5*(W(2,J,1)+W(2,J,KN+1))
        T(1,J,K)=0.5*(T(2,J,1)+T(2,J,KN+1))
        IF(T(1,J,K).LE.RTINFA)XMU(1,J,K)=T(1,J,K)
        IF(T(1,J,K).GT.RTINFA)XMU(1,J,K)=
       & T(1,J,K)*230.0*SQRT(T(1,J,K)*TINFA)/(T(1,J,K)*TINF+110.0)
  100   CONTINUE
        RETURN
        END
```

4.7 Subroutine ROS3D

This subroutine is the core for the algorithm. The EQUIVALENCE technique is used here again for increasing the efficiency of the interpolations of the left and right column hyper-vectors of the primitive variables for invisid flux evaluations. The PVR array is also used in evaluating the diffusive fluxes. INCLUDE statement is used here not only for inserting the COMMON declarations but also for inserting the Osher's approximate solver and diffusive fluxes evaluation modules. The file "osher.f" can be shared with TPNS3D. The file can also be replaced by a file of the Roe's approximate solver, or a file of O-variant Osher's approximate solver, which we did for the code validation. It is interesting to compare the convergence rates of these three solvers.

```
      SUBROUTINE ROS3D
      INCLUDE "comm.f"
      COMMON /STATE/ RHO(IM,JM,KM),U(IM,JM,KM),V(IM,JM,KM),W(IM,JM,KM)
     &          ,T(IM,JM,KM),XMU(IM,JM,KM)
      DIMENSION PVR(IM,JM,KM,5)
      DIMENSION QL(5),QR(5),EIB(5),FIB(5),PM(5,5),EIG(5),DOM(5)
      DIMENSION E(5),F(5),G(5)
      DIMENSION SXII(3,2),SETAI(3,2),SZETI(3,2)
      DIMENSION SXIJ(3,2),SETAJ(3,2),SZETJ(3,2)
      DIMENSION SXIK(3,2),SETAK(3,2),SZETK(3,2)
      EQUIVALENCE (RHO(1,1,1),PVR(1,1,1,1))
C
C --- flux in xi direction, EWAVE at (i+1/2,j,k)
      DO 1000 K=2,KN
      DO 1000 J=2,JN
      DO 1000 I=2,IN-1
C
      DO 10 L=1,5
      DP=PVR(I+1,J,K,L)-PVR(I,J,K,L)
      IF(I.EQ.2)THEN
       DM=2.0*(PVR(2,J,K,L)-PVR(1,J,K,L))
      ELSE
       DM=PVR(I,J,K,L)-PVR(I-1,J,K,L)
      ENDIF
      S=(2.0*DP*DM+EPS)/(DP*DP+DM*DM+EPS)
      QL(L)=PVR(I,J,K,L)+0.25*S*((1.0-CK*S)*DM+(1.0+CK*S)*DP)
      DM=PVR(I+1,J,K,L)-PVR(I,J,K,L)
      IF(I.EQ.IN-1)THEN
       DP=2.0*(PVR(I+2,J,K,L)-PVR(I+1,J,K,L))
      ELSE
       DP=PVR(I+2,J,K,L)-PVR(I+1,J,K,L)
      ENDIF
      S=(2.0*DP*DM+EPS)/(DP*DP+DM*DM+EPS)
      QR(L)=PVR(I+1,J,K,L)-0.25*S*((1.0+CK*S)*DM+(1.0-CK*S)*DP)
 10   CONTINUE
C
      AS=AREA(I,J,K,1)
      SS=1.0/AS
      C1=SXI(I,J,K,1)*SS
      C2=SXI(I,J,K,2)*SS
      C3=SXI(I,J,K,3)*SS
C
C --- Roe's Riemann solver
C
C     INCLUDE "roe.f"
C
C --- Osher's Riemann solver
C
      INCLUDE "../ns3dt/osher.f"
C
C --   VISCOUS TERM IN X-DIRECTION ----------------
C This part can be replaced by diff3d.f -- until next C
      C1=0.5*(XMU(I,J,K)+XMU(I+1,J,K)+XMUT(I,J,K)+XMUT(I+1,J,K))*RRE
      C2=2.0*C1/3.0
      C3=(XMU(I,J,K)+XMU(I+1,J,K))*C0+(XMUT(I,J,K)+XMUT(I+1,J,K))*CT
      SXII(1,1)=0.5*(SXI(I,J,K,1)+SXI(I-1,J,K,1))
      SXII(1,2)=0.5*(SXI(I,J,K,1)+SXI(I+1,J,K,1))
```

```
      SETAI(1,1)=0.5*(SETA(I,J-1,K,1)+SETA(I+1,J-1,K,1))
      SETAI(1,2)=0.5*(SETA(I,J,K,1)+SETA(I+1,J,K,1))
      SZETI(1,1)=0.5*(SZET(I,J,K-1,1)+SZET(I+1,J,K-1,1))
      SZETI(1,2)=0.5*(SZET(I,J,K,1)+SZET(I+1,J,K,1))
      SXII(2,1)=0.5*(SXI(I,J,K,2)+SXI(I-1,J,K,2))
      SXII(2,2)=0.5*(SXI(I,J,K,2)+SXI(I+1,J,K,2))
      SETAI(2,1)=0.5*(SETA(I,J-1,K,2)+SETA(I+1,J-1,K,2))
      SETAI(2,2)=0.5*(SETA(I,J,K,2)+SETA(I+1,J,K,2))
      SZETI(2,1)=0.5*(SZET(I,J,K-1,2)+SZET(I+1,J,K-1,2))
      SZETI(2,2)=0.5*(SZET(I,J,K,2)+SZET(I+1,J,K,2))
      SXII(3,1)=0.5*(SXI(I,J,K,3)+SXI(I-1,J,K,3))
      SXII(3,2)=0.5*(SXI(I,J,K,3)+SXI(I+1,J,K,3))
      SETAI(3,1)=0.5*(SETA(I,J-1,K,3)+SETA(I+1,J-1,K,3))
      SETAI(3,2)=0.5*(SETA(I,J,K,3)+SETA(I+1,J,K,3))
      SZETI(3,1)=0.5*(SZET(I,J,K-1,3)+SZET(I+1,J,K-1,3))
      SZETI(3,2)=0.5*(SZET(I,J,K,3)+SZET(I+1,J,K,3))
      UM=0.5*(U(I+1,J,K)+U(I,J,K))
      VM=0.5*(V(I+1,J,K)+V(I,J,K))
      WM=0.5*(W(I+1,J,K)+W(I,J,K))
      UM11=0.25*(U(I+1,J-1,K)+U(I,J-1,K)+U(I+1,J,K)+U(I,J,K))
      UM12=0.25*(U(I+1,J+1,K)+U(I,J+1,K)+U(I+1,J,K)+U(I,J,K))
      UM21=0.25*(U(I+1,J,K-1)+U(I,J,K-1)+U(I+1,J,K)+U(I,J,K))
      UM22=0.25*(U(I+1,J,K+1)+U(I,J,K+1)+U(I+1,J,K)+U(I,J,K))
      VM11=0.25*(V(I+1,J-1,K)+V(I,J-1,K)+V(I+1,J,K)+V(I,J,K))
      VM12=0.25*(V(I+1,J+1,K)+V(I,J+1,K)+V(I+1,J,K)+V(I,J,K))
      VM21=0.25*(V(I+1,J,K-1)+V(I,J,K-1)+V(I+1,J,K)+V(I,J,K))
      VM22=0.25*(V(I+1,J,K+1)+V(I,J,K+1)+V(I+1,J,K)+V(I,J,K))
      WM11=0.25*(W(I+1,J-1,K)+W(I,J-1,K)+W(I+1,J,K)+W(I,J,K))
      WM12=0.25*(W(I+1,J+1,K)+W(I,J+1,K)+W(I+1,J,K)+W(I,J,K))
      WM21=0.25*(W(I+1,J,K-1)+W(I,J,K-1)+W(I+1,J,K)+W(I,J,K))
      WM22=0.25*(W(I+1,J,K+1)+W(I,J,K+1)+W(I+1,J,K)+W(I,J,K))
      TM11=0.25*(T(I+1,J-1,K)+T(I,J-1,K)+T(I+1,J,K)+T(I,J,K))
      TM12=0.25*(T(I+1,J+1,K)+T(I,J+1,K)+T(I+1,J,K)+T(I,J,K))
      TM21=0.25*(T(I+1,J,K-1)+T(I,J,K-1)+T(I+1,J,K)+T(I,J,K))
      TM22=0.25*(T(I+1,J,K+1)+T(I,J,K+1)+T(I+1,J,K)+T(I,J,K))
      RECVOLM=2.0/(VOL(I,J,K)+VOL(I+1,J,K))
      PUPX=(SXII (1,2)*U(I+1,J,K)-SXII (1,1)*U(I,J,K)
     &     +SETAI(1,2)*UM12-SETAI(1,1)*UM11
     &     +SZETI(1,2)*UM22-SZETI(1,1)*UM21)*RECVOLM
      PUPY=(SXII (2,2)*U(I+1,J,K)-SXII (2,1)*U(I,J,K)
     &     +SETAI(2,2)*UM12-SETAI(2,1)*UM11
     &     +SZETI(2,2)*UM22-SZETI(2,1)*UM21)*RECVOLM
      PUPZ=(SXII (3,2)*U(I+1,J,K)-SXII (3,1)*U(I,J,K)
     &     +SETAI(3,2)*UM12-SETAI(3,1)*UM11
     &     +SZETI(3,2)*UM22-SZETI(3,1)*UM21)*RECVOLM
      PVPX=(SXII (1,2)*V(I+1,J,K)-SXII (1,1)*V(I,J,K)
     &     +SETAI(1,2)*VM12-SETAI(1,1)*VM11
     &     +SZETI(1,2)*VM22-SZETI(1,1)*VM21)*RECVOLM
      PVPY=(SXII (2,2)*V(I+1,J,K)-SXII (2,1)*V(I,J,K)
     &     +SETAI(2,2)*VM12-SETAI(2,1)*VM11
     &     +SZETI(2,2)*VM22-SZETI(2,1)*VM21)*RECVOLM
      PVPZ=(SXII (3,2)*V(I+1,J,K)-SXII (3,1)*V(I,J,K)
     &     +SETAI(3,2)*VM12-SETAI(3,1)*VM11
     &     +SZETI(3,2)*VM22-SZETI(3,1)*VM21)*RECVOLM
      PWPX=(SXII (1,2)*W(I+1,J,K)-SXII (1,1)*W(I,J,K)
     &     +SETAI(1,2)*WM12-SETAI(1,1)*WM11
     &     +SZETI(1,2)*WM22-SZETI(1,1)*WM21)*RECVOLM
      PWPY=(SXII (2,2)*W(I+1,J,K)-SXII (2,1)*W(I,J,K)
```

```
     &     +SETAI(2,2)*WM12-SETAI(2,1)*WM11
     &     +SZETI(2,2)*WM22-SZETI(2,1)*WM21)*RECVOLM
       PWPZ=(SXII (3,2)*W(I+1,J,K)-SXII (3,1)*W(I,J,K)
     &     +SETAI(3,2)*WM12-SETAI(3,1)*WM11
     &     +SZETI(3,2)*TM22-SZETI(3,1)*TM21)*RECVOLM
       PTPX=(SXII (1,2)*T(I+1,J,K)-SXII (1,1)*T(I,J,K)
     &     +SETAI(1,2)*TM12-SETAI(1,1)*TM11
     &     +SZETI(1,2)*TM22-SZETI(1,1)*TM21)*RECVOLM
       PTPY=(SXII (2,2)*T(I+1,J,K)-SXII (2,1)*T(I,J,K)
     &     +SETAI(2,2)*TM12-SETAI(2,1)*TM11
     &     +SZETI(2,2)*TM22-SZETI(2,1)*TM21)*RECVOLM
       PTPZ=(SXII (3,2)*T(I+1,J,K)-SXII (3,1)*T(I,J,K)
     &     +SETAI(3,2)*TM12-SETAI(3,1)*TM11
     &     +SZETI(3,2)*TM22-SZETI(3,1)*TM21)*RECVOLM
       TAUXX=C2*(2.0*PUPX-PVPY-PWPZ)
       TAUYY=C2*(2.0*PVPY-PUPX-PWPZ)
       TAUZZ=C2*(2.0*PWPZ-PUPX-PVPY)
       TAUXY=C1*(PUPY+PVPX)
       TAUXZ=C1*(PUPZ+PWPX)
       TAUYZ=C1*(PVPZ+PWPY)
       E(2)=TAUXX
       E(3)=TAUXY
       E(4)=TAUXZ
       E(5)=UM*TAUXX+VM*TAUXY+WM*TAUXZ+C3*PTPX
       F(2)=TAUXY
       F(3)=TAUYY
       F(4)=TAUYZ
       F(5)=UM*TAUXY+VM*TAUYY+WM*TAUYZ+C3*PTPY
       G(2)=TAUXZ
       G(3)=TAUYZ
       G(4)=TAUZZ
       G(5)=UM*TAUXZ+VM*TAUYZ+WM*TAUZZ+C3*PTPZ
C
       FIB(2)=FIB(2)-(SXI(I,J,K,1)*E(2)+SXI(I,J,K,2)*F(2)
     &              +SXI(I,J,K,3)*G(2))
       FIB(3)=FIB(3)-(SXI(I,J,K,1)*E(3)+SXI(I,J,K,2)*F(3)
     &              +SXI(I,J,K,3)*G(3))
       FIB(4)=FIB(4)-(SXI(I,J,K,1)*E(4)+SXI(I,J,K,2)*F(4)
     &              +SXI(I,J,K,3)*G(4))
       FIB(5)=FIB(5)-(SXI(I,J,K,1)*E(5)+SXI(I,J,K,2)*F(5)
     &              +SXI(I,J,K,3)*G(5))
       RR(I,J,K,1)=RR(I,J,K,1)+FIB(1)
       RR(I,J,K,2)=RR(I,J,K,2)+FIB(2)
       RR(I,J,K,3)=RR(I,J,K,3)+FIB(3)
       RR(I,J,K,4)=RR(I,J,K,4)+FIB(4)
       RR(I,J,K,5)=RR(I,J,K,5)+FIB(5)
       RR(I+1,J,K,1)=RR(I+1,J,K,1)-FIB(1)
       RR(I+1,J,K,2)=RR(I+1,J,K,2)-FIB(2)
       RR(I+1,J,K,3)=RR(I+1,J,K,3)-FIB(3)
       RR(I+1,J,K,4)=RR(I+1,J,K,4)-FIB(4)
       RR(I+1,J,K,5)=RR(I+1,J,K,5)-FIB(5)
C
1000   CONTINUE
C
C --- flux in eta direction, FWAVE at (i,j+1/2,k)
C
       DO 3000 K=2,KN
       DO 3000 J=2,JN-1
```

```
      DO 3000 I=2,IN
C
      DO 30 L=1,5
       DP=PVR(I,J+1,K,L)-PVR(I,J,K,L)
      IF(J.EQ.2)THEN
       DM=2.0*(PVR(I,2,K,L)-PVR(I,1,K,L))
      ELSE
      DM=PVR(I,J,K,L)-PVR(I,J-1,K,L)
      ENDIF
      S=(2.0*DP*DM+EPS)/(DP*DP+DM*DM+EPS)
      QL(L)=PVR(I,J,K,L)+0.25*S*((1.0-CK*S)*DM+(1.0+CK*S)*DP)
      DM=PVR(I,J+1,K,L)-PVR(I,J,K,L)
      IF(J.EQ.JN-1)THEN
       DP=2.0*(PVR(I,JN+1,K,L)-PVR(I,JN,K,L))
      ELSE
       DP=PVR(I,J+2,K,L)-PVR(I,J+1,K,L)
      ENDIF
      S=(2.0*DP*DM+EPS)/(DP*DP+DM*DM+EPS)
      QR(L)=PVR(I,J+1,K,L)-0.25*S*((1.0+CK*S)*DM+(1.0-CK*S)*DP)
 30   CONTINUE
C
      AS=AREA(I,J,K,2)
      SS=1.0/AS
      C1=SETA(I,J,K,1)*SS
      C2=SETA(I,J,K,2)*SS
      C3=SETA(I,J,K,3)*SS
C
C --- Roe's Riemann solver
C
C     INCLUDE "roe.f"
C
C --- Osher's Riemann solver
C
      INCLUDE "../ns3dt/osher.f"
C ------ Viscous term in eta direction -----------------
C  This part can be replaced by diff3d.f -- until next C
      C1=0.5*(XMU(I,J,K)+XMU(I,J+1,K)+XMUT(I,J,K)+XMUT(I,J+1,K))*RRE
      C2=2.0*C1/3.0
      C3=(XMU(I,J,K)+XMU(I,J+1,K))*C0+(XMUT(I,J,K)+XMUT(I,J+1,K))*CT
      SXIJ (1,1)=0.5*(SXI (I-1,J,K,1)+SXI(I-1,J+1,K,1))
      SXIJ (1,2)=0.5*(SXI (I,J,K,1)+SXI(I,J+1,K,1))
      SETAJ(1,1)=0.5*(SETA(I,J-1,K,1)+SETA(I,J,K,1))
      SETAJ(1,2)=0.5*(SETA(I,J+1,K,1)+SETA(I,J,K,1))
      SZETJ(1,1)=0.5*(SZET(I,J,K-1,1)+SZET(I,J+1,K-1,1))
      SZETJ(1,2)=0.5*(SZET(I,J,K,1)+SZET(I,J+1,K,1))
      SXIJ (2,1)=0.5*(SXI (I-1,J,K,2)+SXI(I-1,J+1,K,2))
      SXIJ (2,2)=0.5*(SXI (I,J,K,2)+SXI(I,J+1,K,2))
      SETAJ(2,1)=0.5*(SETA(I,J-1,K,2)+SETA(I,J,K,2))
      SETAJ(2,2)=0.5*(SETA(I,J+1,K,2)+SETA(I,J,K,2))
      SZETJ(2,1)=0.5*(SZET(I,J,K-1,2)+SZET(I,J+1,K-1,2))
      SZETJ(2,2)=0.5*(SZET(I,J,K,2)+SZET(I,J+1,K,2))
      SXIJ (3,1)=0.5*(SXI (I-1,J,K,3)+SXI(I-1,J+1,K,3))
      SXIJ (3,2)=0.5*(SXI (I,J,K,3)+SXI(I,J+1,K,3))
      SETAJ(3,1)=0.5*(SETA(I,J-1,K,3)+SETA(I,J,K,3))
      SETAJ(3,2)=0.5*(SETA(I,J+1,K,3)+SETA(I,J,K,3))
      SZETJ(3,1)=0.5*(SZET(I,J,K-1,3)+SZET(I,J+1,K-1,3))
      SZETJ(3,2)=0.5*(SZET(I,J,K,3)+SZET(I,J+1,K,3))
      UM=0.5*(U(I,J+1,K)+U(I,J,K))
```

```
      VM=0.5*(V(I,J+1,K)+V(I,J,K))
      WM=0.5*(W(I,J+1,K)+W(I,J,K))
      UM11=0.25*(U(I-1,J+1,K)+U(I-1,J,K)+U(I,J+1,K)+U(I,J,K))
      UM12=0.25*(U(I+1,J+1,K)+U(I+1,J,K)+U(I,J+1,K)+U(I,J,K))
      UM21=0.25*(U(I,J+1,K-1)+U(I,J,K-1)+U(I,J+1,K)+U(I,J,K))
      UM22=0.25*(U(I,J+1,K+1)+U(I,J,K+1)+U(I,J+1,K)+U(I,J,K))
      VM11=0.25*(V(I-1,J+1,K)+V(I-1,J,K)+V(I,J+1,K)+V(I,J,K))
      VM12=0.25*(V(I+1,J+1,K)+V(I+1,J,K)+V(I,J+1,K)+V(I,J,K))
      VM21=0.25*(V(I,J+1,K-1)+V(I,J,K-1)+V(I,J+1,K)+V(I,J,K))
      VM22=0.25*(V(I,J+1,K+1)+V(I,J,K+1)+V(I,J+1,K)+V(I,J,K))
      WM11=0.25*(W(I-1,J+1,K)+W(I-1,J,K)+W(I,J+1,K)+W(I,J,K))
      WM12=0.25*(W(I+1,J+1,K)+W(I+1,J,K)+W(I,J+1,K)+W(I,J,K))
      WM21=0.25*(W(I,J+1,K-1)+W(I,J,K-1)+W(I,J+1,K)+W(I,J,K))
      WM22=0.25*(W(I,J+1,K+1)+W(I,J,K+1)+W(I,J+1,K)+W(I,J,K))
      TM11=0.25*(T(I-1,J+1,K)+T(I-1,J,K)+T(I,J+1,K)+T(I,J,K))
      TM12=0.25*(T(I+1,J+1,K)+T(I+1,J,K)+T(I,J+1,K)+T(I,J,K))
      TM21=0.25*(T(I,J+1,K-1)+T(I,J,K-1)+T(I,J+1,K)+T(I,J,K))
      TM22=0.25*(T(I,J+1,K+1)+T(I,J,K+1)+T(I,J+1,K)+T(I,J,K))
      RECVOLM=2.0/(VOL(I,J,K)+VOL(I,J+1,K))
      PUPX=(SETAJ(1,2)*U(I,J+1,K)-SETAJ(1,1)*U(I,J,K)
     &     +SXIJ (1,2)*UM12-SXIJ (1,1)*UM11
     &     +SZETJ(1,2)*UM22-SZETJ(1,1)*UM21)*RECVOLM
      PUPY=(SETAJ(2,2)*U(I,J+1,K)-SETAJ(2,1)*U(I,J,K)
     &     +SXIJ (2,2)*UM12-SXIJ (2,1)*UM11
     &     +SZETJ(2,2)*UM22-SZETJ(2,1)*UM21)*RECVOLM
      PUPZ=(SETAJ(3,2)*U(I,J+1,K)-SETAJ(3,1)*U(I,J,K)
     &     +SXIJ (3,2)*UM12-SXIJ (3,1)*UM11
     &     +SZETJ(3,2)*UM22-SZETJ(3,1)*UM21)*RECVOLM
      PVPX=(SETAJ(1,2)*V(I,J+1,K)-SETAJ(1,1)*V(I,J,K)
     &     +SXIJ (1,2)*VM12-SXIJ (1,1)*VM11
     &     +SZETJ(1,2)*VM22-SZETJ(1,1)*VM21)*RECVOLM
      PVPY=(SETAJ(2,2)*V(I,J+1,K)-SETAJ(2,1)*V(I,J,K)
     &     +SXIJ (2,2)*VM12-SXIJ (2,1)*VM11
     &     +SZETJ(2,2)*VM22-SZETJ(2,1)*VM21)*RECVOLM
      PVPZ=(SETAJ(3,2)*V(I,J+1,K)-SETAJ(3,1)*V(I,J,K)
     &     +SXIJ (3,2)*VM12-SXIJ (3,1)*VM11
     &     +SZETJ(3,2)*VM22-SZETJ(3,1)*VM21)*RECVOLM
      PWPX=(SETAJ(1,2)*W(I,J+1,K)-SETAJ(1,1)*W(I,J,K)
     &     +SXIJ (1,2)*WM12-SXIJ (1,1)*WM11
     &     +SZETJ(1,2)*WM22-SZETJ(1,1)*WM21)*RECVOLM
      PWPY=(SETAJ(2,2)*W(I,J+1,K)-SETAJ(2,1)*W(I,J,K)
     &     +SXIJ (2,2)*WM12-SXIJ (2,1)*WM11
     &     +SZETJ(2,2)*WM22-SZETJ(2,1)*WM21)*RECVOLM
      PWPZ=(SETAJ(3,2)*W(I,J+1,K)-SETAJ(3,1)*W(I,J,K)
     &     +SXIJ (3,2)*WM12-SXIJ (3,1)*WM11
     &     +SZETJ(3,2)*WM22-SZETJ(3,1)*WM21)*RECVOLM
      PTPX=(SETAJ(1,2)*T(I,J+1,K)-SETAJ(1,1)*T(I,J,K)
     &     +SXIJ (1,2)*TM12-SXIJ (1,1)*TM11
     &     +SZETJ(1,2)*TM22-SZETJ(1,1)*TM21)*RECVOLM
      PTPY=(SETAJ(2,2)*T(I,J+1,K)-SETAJ(2,1)*T(I,J,K)
     &     +SXIJ (2,2)*TM12-SXIJ (2,1)*TM11
     &     +SZETJ(2,2)*TM22-SZETJ(2,1)*TM21)*RECVOLM
      PTPZ=(SETAJ(3,2)*T(I,J+1,K)-SETAJ(3,1)*T(I,J,K)
     &     +SXIJ (3,2)*TM12-SXIJ (3,1)*TM11
     &     +SZETJ(3,2)*TM22-SZETJ(3,1)*TM21)*RECVOLM
      TAUXX=C2*(2.0*PUPX-PVPY-PWPZ)
      TAUYY=C2*(2.0*PVPY-PUPX-PWPZ)
      TAUZZ=C2*(2.0*PWPZ-PUPX-PVPY)
```

```
          TAUXY=C1*(PUPY+PVPX)
          TAUXZ=C1*(PUPZ+PWPX)
          TAUYZ=C1*(PVPZ+PWPY)
          E(2)=TAUXX
          E(3)=TAUXY
          E(4)=TAUXZ
          E(5)=UM*TAUXX+VM*TAUXY+WM*TAUXZ+C3*PTPX
          F(2)=TAUXY
          F(3)=TAUYY
          F(4)=TAUYZ
          F(5)=UM*TAUXY+VM*TAUYY+WM*TAUYZ+C3*PTPY
          G(2)=TAUXZ
          G(3)=TAUYZ
          G(4)=TAUZZ
          G(5)=UM*TAUXZ+VM*TAUYZ+WM*TAUZZ+C3*PTPZ
C
          FIB(2)=FIB(2)-(SETA(I,J,K,1)*E(2)+SETA(I,J,K,2)*F(2)
         &              +SETA(I,J,K,3)*G(2))
          FIB(3)=FIB(3)-(SETA(I,J,K,1)*E(3)+SETA(I,J,K,2)*F(3)
         &              +SETA(I,J,K,3)*G(3))
          FIB(4)=FIB(4)-(SETA(I,J,K,1)*E(4)+SETA(I,J,K,2)*F(4)
         &              +SETA(I,J,K,3)*G(4))
          FIB(5)=FIB(5)-(SETA(I,J,K,1)*E(5)+SETA(I,J,K,2)*F(5)
         &              +SETA(I,J,K,3)*G(5))
          RR(I,J,K,1)=RR(I,J,K,1)+FIB(1)
          RR(I,J,K,2)=RR(I,J,K,2)+FIB(2)
          RR(I,J,K,3)=RR(I,J,K,3)+FIB(3)
          RR(I,J,K,4)=RR(I,J,K,4)+FIB(4)
          RR(I,J,K,5)=RR(I,J,K,5)+FIB(5)
          RR(I,J+1,K,1)=RR(I,J+1,K,1)-FIB(1)
          RR(I,J+1,K,2)=RR(I,J+1,K,2)-FIB(2)
          RR(I,J+1,K,3)=RR(I,J+1,K,3)-FIB(3)
          RR(I,J+1,K,4)=RR(I,J+1,K,4)-FIB(4)
          RR(I,J+1,K,5)=RR(I,J+1,K,5)-FIB(5)
3000      CONTINUE
C
C --- flux in zet direction, GWAVE at (i,j,k+1/2)
C
          DO 5000 K=2,KN-1
          DO 5000 J=2,JN
          DO 5000 I=2,IN
C
          DO 50 L=1,5
          DP=PVR(I,J,K+1,L)-PVR(I,J,K,L)
          DM=PVR(I,J,K,L)-PVR(I,J,K-1,L)
          S=(2.0*DP*DM+EPS)/(DP*DP+DM*DM+EPS)
          QL(L)=PVR(I,J,K,L)+0.25*S*((1.0-CK*S)*DM+(1.0+CK*S)*DP)
          DM=PVR(I,J,K+1,L)-PVR(I,J,K,L)
          DP=PVR(I,J,K+2,L)-PVR(I,J,K+1,L)
          S=(2.0*DP*DM+EPS)/(DP*DP+DM*DM+EPS)
          QR(L)=PVR(I,J,K+1,L)-0.25*S*((1.0+CK*S)*DM+(1.0-CK*S)*DP)
 50       CONTINUE
C
          AS=AREA(I,J,K,3)
          SS=1.0/AS
          C1=SZET(I,J,K,1)*SS
          C2=SZET(I,J,K,2)*SS
          C3=SZET(I,J,K,3)*SS
```

```
C
C --- Roe's Riemann solver
C
C    INCLUDE "roe.f"
C
C --- Osher's Riemann solver
C
        INCLUDE "osher.f"
C  ---- VISCOUS TERM IN Z-DIRECTION -----------
C  This part can be replaced by diff3d.f -- until next C
          C1=0.5*(XMU(I,J,K)+XMU(I,J,K+1)+XMUT(I,J,K)+XMUT(I,J,K+1))*RRE
          C2=2.0*C1/3.0
          C3=(XMU(I,J,K)+XMU(I,J,K+1))*C0+(XMUT(I,J,K)+XMUT(I,J,K+1))*CT
          SXIK(1,1)=0.5*(SXI(I-1,J,K,1)+SXI(I-1,J,K+1,1))
          SXIK(1,2)=0.5*(SXI(I,J,K,1)+SXI(I,J,K+1,1))
          SETAK(1,1)=0.5*(SETA(I,J-1,K,1)+SETA(I,J-1,K+1,1))
          SETAK(1,2)=0.5*(SETA(I,J,K,1)+SETA(I,J,K+1,1))
          SZETK(1,1)=0.5*(SZET(I,J,K,1)+SZET(I,J,K-1,1))
          SZETK(1,2)=0.5*(SZET(I,J,K,1)+SZET(I,J,K+1,1))
          SXIK(2,1)=0.5*(SXI(I-1,J,K,2)+SXI(I-1,J,K+1,2))
          SXIK(2,2)=0.5*(SXI(I,J,K,2)+SXI(I,J,K+1,2))
          SETAK(2,1)=0.5*(SETA(I,J-1,K,2)+SETA(I,J-1,K+1,2))
          SETAK(2,2)=0.5*(SETA(I,J,K,2)+SETA(I,J,K+1,2))
          SZETK(2,1)=0.5*(SZET(I,J,K,2)+SZET(I,J,K-1,2))
          SZETK(2,2)=0.5*(SZET(I,J,K,2)+SZET(I,J,K+1,2))
          SXIK(3,1)=0.5*(SXI(I-1,J,K,3)+SXI(I-1,J,K+1,3))
          SXIK(3,2)=0.5*(SXI(I,J,K,3)+SXI(I,J,K+1,3))
          SETAK(3,1)=0.5*(SETA(I,J-1,K,3)+SETA(I,J-1,K+1,3))
          SETAK(3,2)=0.5*(SETA(I,J,K,3)+SETA(I,J,K+1,3))
          SZETK(3,1)=0.5*(SZET(I,J,K,3)+SZET(I,J,K-1,3))
          SZETK(3,2)=0.5*(SZET(I,J,K,3)+SZET(I,J,K+1,3))
          UM=0.5*(U(I,J,K+1)+U(I,J,K))
          VM=0.5*(V(I,J,K+1)+V(I,J,K))
          WM=0.5*(W(I,J,K+1)+W(I,J,K))
          UM11=0.25*(U(I-1,J,K+1)+U(I,J,K+1)+U(I-1,J,K)+U(I,J,K))
          UM12=0.25*(U(I+1,J,K+1)+U(I,J,K+1)+U(I+1,J,K)+U(I,J,K))
          UM21=0.25*(U(I,J-1,K+1)+U(I,J-1,K)+U(I,J,K+1)+U(I,J,K))
          UM22=0.25*(U(I,J+1,K+1)+U(I,J+1,K)+U(I,J,K+1)+U(I,J,K))
          VM11=0.25*(V(I-1,J,K+1)+V(I,J,K+1)+V(I-1,J,K)+V(I,J,K))
          VM12=0.25*(V(I+1,J,K+1)+V(I,J,K+1)+V(I+1,J,K)+V(I,J,K))
          VM21=0.25*(V(I,J-1,K+1)+V(I,J-1,K)+V(I,J,K+1)+V(I,J,K))
          VM22=0.25*(V(I,J+1,K+1)+V(I,J+1,K)+V(I,J,K+1)+V(I,J,K))
          WM11=0.25*(W(I-1,J,K+1)+W(I,J,K+1)+W(I-1,J,K)+W(I,J,K))
          WM12=0.25*(W(I+1,J,K+1)+W(I,J,K+1)+W(I+1,J,K)+W(I,J,K))
          WM21=0.25*(W(I,J-1,K+1)+W(I,J-1,K)+W(I,J,K+1)+W(I,J,K))
          WM22=0.25*(W(I,J+1,K+1)+W(I,J+1,K)+W(I,J,K+1)+W(I,J,K))
          TM11=0.25*(T(I-1,J,K+1)+T(I,J,K+1)+T(I-1,J,K)+T(I,J,K))
          TM12=0.25*(T(I+1,J,K+1)+T(I,J,K+1)+T(I+1,J,K)+T(I,J,K))
          TM21=0.25*(T(I,J-1,K+1)+T(I,J-1,K)+T(I,J,K+1)+T(I,J,K))
          TM22=0.25*(T(I,J+1,K+1)+T(I,J+1,K)+T(I,J,K+1)+T(I,J,K))
          RECVOLM=2.0/(VOL(I,J,K)+VOL(I,J,K+1))
          PUPX=(SZETK(1,2)*U(I,J,K+1)-SZETK(1,1)*U(I,J,K)
         &     +SETAK(1,2)*UM22-SETAK(1,1)*UM21
         &     +SXIK (1,2)*UM12-SXIK (1,1)*UM11)*RECVOLM
          PUPY=(SZETK(2,2)*U(I,J,K+1)-SZETK(2,1)*U(I,J,K)
         &     +SETAK(2,2)*UM22-SETAK(2,1)*UM21
         &     +SXIK (2,2)*UM12-SXIK (2,1)*UM11)*RECVOLM
          PUPZ=(SZETK(3,2)*U(I,J,K+1)-SZETK(3,1)*U(I,J,K)
```

```
     &    +SETAK(3,2)*UM22-SETAK(3,1)*UM21
     &    +SXIK (3,2)*UM12-SXIK (3,1)*UM11)*RECVOLM
          PVPX=(SZETK(1,2)*V(I,J,K+1)-SZETK(1,1)*V(I,J,K)
     &    +SETAK(1,2)*VM22-SETAK(1,1)*VM21
     &    +SXIK (1,2)*VM12-SXIK (1,1)*VM11)*RECVOLM
          PVPY=(SZETK(2,2)*V(I,J,K+1)-SZETK(2,1)*V(I,J,K)
     &    +SETAK(2,2)*VM22-SETAK(2,1)*VM21
     &    +SXIK (2,2)*VM12-SXIK (2,1)*VM11)*RECVOLM
          PVPZ=(SZETK(3,2)*V(I,J,K+1)-SZETK(3,1)*V(I,J,K)
     &    +SETAK(3,2)*VM22-SETAK(3,1)*VM21
     &    +SXIK (3,2)*VM12-SXIK (3,1)*VM11)*RECVOLM
          PWPX=(SZETK(1,2)*W(I,J,K+1)-SZETK(1,1)*W(I,J,K)
     &    +SETAK(1,2)*WM22-SETAK(1,1)*WM21
     &    +SXIK (1,2)*WM12-SXIK (1,1)*WM11)*RECVOLM
          PWPY=(SZETK(2,2)*W(I,J,K+1)-SZETK(2,1)*W(I,J,K)
     &    +SETAK(2,2)*WM22-SETAK(2,1)*WM21
     &    +SXIK (2,2)*WM12-SXIK (2,1)*WM11)*RECVOLM
          PWPZ=(SZETK(3,2)*W(I,J,K+1)-SZETK(3,1)*W(I,J,K)
     &    +SETAK(3,2)*WM22-SETAK(3,1)*WM21
     &    +SXIK (3,2)*WM12-SXIK (3,1)*WM11)*RECVOLM
          PTPX=(SZETK(1,2)*T(I,J,K+1)-SZETK(1,1)*T(I,J,K)
     &    +SETAK(1,2)*TM22-SETAK(1,1)*TM21
     &    +SXIK (1,2)*TM12-SXIK (1,1)*TM11)*RECVOLM
          PTPY=(SZETK(2,2)*T(I,J,K+1)-SZETK(2,1)*T(I,J,K)
     &    +SETAK(2,2)*TM22-SETAK(2,1)*TM21
     &    +SXIK (2,2)*TM12-SXIK (2,1)*TM11)*RECVOLM
          PTPZ=(SZETK(3,2)*T(I,J,K+1)-SZETK(3,1)*T(I,J,K)
     &    +SETAK(3,2)*TM22-SETAK(3,1)*TM21
     &    +SXIK (3,2)*TM12-SXIK (3,1)*TM11)*RECVOLM
          TAUXX=C2*(2.0*PUPX-PVPY-PWPZ)
          TAUYY=C2*(2.0*PVPY-PUPX-PWPZ)
          TAUZZ=C2*(2.0*PWPZ-PUPX-PVPY)
          TAUXY=C1*(PUPY+PVPX)
          TAUXZ=C1*(PUPZ+PWPX)
          TAUYZ=C1*(PVPZ+PWPY)
          E(2)=TAUXX
          E(3)=TAUXY
          E(4)=TAUXZ
          E(5)=UM*TAUXX+VM*TAUXY+WM*TAUXZ+C3*PTPX
          F(2)=TAUXY
          F(3)=TAUYY
          F(4)=TAUYZ
          F(5)=UM*TAUXY+VM*TAUYY+WM*TAUYZ+C3*PTPY
          G(2)=TAUXZ
          G(3)=TAUYZ
          G(4)=TAUZZ
          G(5)=UM*TAUXZ+VM*TAUYZ+WM*TAUZZ+C3*PTPZ
C
          FIB(2)=FIB(2)-(SZET(I,J,K,1)*E(2)+SZET(I,J,K,2)*F(2)
     &                  +SZET(I,J,K,3)*G(2))
          FIB(3)=FIB(3)-(SZET(I,J,K,1)*E(3)+SZET(I,J,K,2)*F(3)
     &                  +SZET(I,J,K,3)*G(3))
          FIB(4)=FIB(4)-(SZET(I,J,K,1)*E(4)+SZET(I,J,K,2)*F(4)
     &                  +SZET(I,J,K,3)*G(4))
          FIB(5)=FIB(5)-(SZET(I,J,K,1)*E(5)+SZET(I,J,K,2)*F(5)
     &                  +SZET(I,J,K,3)*G(5))
          RR(I,J,K,1)=RR(I,J,K,1)+FIB(1)
          RR(I,J,K,2)=RR(I,J,K,2)+FIB(2)
```

```
      RR(I,J,K,3)=RR(I,J,K,3)+FIB(3)
      RR(I,J,K,4)=RR(I,J,K,4)+FIB(4)
      RR(I,J,K,5)=RR(I,J,K,5)+FIB(5)
      RR(I,J,K+1,1)=RR(I,J,K+1,1)-FIB(1)
      RR(I,J,K+1,2)=RR(I,J,K+1,2)-FIB(2)
      RR(I,J,K+1,3)=RR(I,J,K+1,3)-FIB(3)
      RR(I,J,K+1,4)=RR(I,J,K+1,4)-FIB(4)
      RR(I,J,K+1,5)=RR(I,J,K+1,5)-FIB(5)
5000  CONTINUE
C
      RETURN
      END
```

4.8 Subroutine UPD3D

This subroutine is designed to update the primitive variables by the local time stepping technique. Also the residuals of the integrations are normmed here for convergence check. The local time step formula is given in section 3.5. The main feature here is that the fifth primitive variable is temperature rather than pressure, thus the non-physical result, such as negative pressure, can be avoided.

```
      SUBROUTINE UPT3D
      INCLUDE "comm.f"
      COMMON /STATE/ RHO(IM,JM,KM),U(IM,JM,KM),V(IM,JM,KM),W(IM,JM,KM)
     &            ,T(IM,JM,KM),XMU(IM,JM,KM)
      R2NORM=0.0
      RMAX=0.0
      UDIF=0.0
      DO 3 K=2,KN
      DO 3 J=2,JN
      DO 3 I=2,IN
C
C-- LOCOL TIME STEP CALCULATION ------------------------------
C
      UU=SXI(I,J,K,1)*U(I,J,K)+SXI(I,J,K,2)*V(I,J,K)
     &  +SXI(I,J,K,3)*W(I,J,K)
      VV=SETA(I,J,K,1)*U(I,J,K)+SETA(I,J,K,2)*V(I,J,K)
     &  +SETA(I,J,K,3)*W(I,J,K)
      WW=SZET(I,J,K,1)*U(I,J,K)+SZET(I,J,K,2)*V(I,J,K)
     &  +SZET(I,J,K,3)*W(I,J,K)
      CS=SQRT(GAM*T(I,J,K)*GAMC6)
      SA1=AREA(I,J,K,1)
      SA2=AREA(I,J,K,2)
      SA3=AREA(I,J,K,3)
      SOUND=(SA1+SA2+SA3)*CS
      REC1=1.0/(RHO(I,J,K)*RE*PR*VOL(I,J,K))
      DTL=CFL/(ABS(UU)+ABS(VV)+ABS(WW)+SOUND+2.0*GAM*REC1*
     &  XMU(I,J,K)*(SA1**2+SA2**2+SA3**2))
C
C-- UPDATING THE VARIABLES ----------
C
      VOS2=0.5*(U(I,J,K)**2+V(I,J,K)**2+W(I,J,K)**2)
      Q1=RHO(I,J,K)
      Q2=Q1*U(I,J,K)
      Q3=Q1*V(I,J,K)
      Q4=Q1*W(I,J,K)
      Q5=Q1*(VOS2+T(I,J,K)*GAMC2*GAMC6)
      Q1=Q1-DTL*RR(I,J,K,1)
```

```
      Q2=Q2-DTL*RR(I,J,K,2)
      Q3=Q3-DTL*RR(I,J,K,3)
      Q4=Q4-DTL*RR(I,J,K,4)
      Q5=Q5-DTL*RR(I,J,K,5)
      RQ1=1.0/Q1
C
      RHO(I,J,K)=Q1
      U(I,J,K)=Q2*RQ1
      V(I,J,K)=Q3*RQ1
      W(I,J,K)=Q4*RQ1
      VOS2=0.5*(U(I,J,K)**2+V(I,J,K)**2+W(I,J,K)**2)
      T(I,J,K)=(Q5*RQ1-VOS2)/(GAMC2*GAMC6)
      UD=2.0*VOS2
      IF(UD.GT.UDIF)UDIF=UD
      IF(T(I,J,K).LE.RTINFA)THEN
      XMU(I,J,K)=T(I,J,K)
      ELSE
      XMU(I,J,K)=T(I,J,K)*230.0*SQRT(T(I,J,K)*TINFA)
     & /(T(I,J,K)*TINF+110.0)
      ENDIF
C
      DO 2 L=1,5
      RD=RR(I,J,K,L)**2
      IF(RD.GE.RMAX) THEN
      RMAX=RD
      IQ=I
      JQ=J
      KQ=K
      LQ=L
      ENDIF
      R2NORM=R2NORM+RD
   2  RR(I,J,K,L)=0.0
C
   3  CONTINUE
      IF(IT.LE.1.AND.IRST.EQ.0)R1NORM=R2NORM
      R3NORM=R2NORM
      R2NORM=SQRT(R2NORM/R1NORM)
      TIME=SECNDS(0.0)
C     CALL SECOND(TIME)
      TIM2=TIME-TIM1
      WRITE(6,9000) IT,R2NORM,R3NORM,IQ,JQ,KQ,LQ,RMAX,TIM2
      WRITE(16,9000) IT,R2NORM,R3NORM,IQ,JQ,KQ,LQ,RMAX,TIM2
      TIM1=TIME
9000  FORMAT(' IT',I5,' R2:',E10.5,' R1:',E11.5,3I4,I2,' RX:',E10.4,
     &' CPU:',E9.3)
      RETURN
      END
```

4.9 Subroutine BLT3D

This is an independent subroutine for the turbulent viscosity evaluation of Baldwin-Lomax model. Removing this subroutine and the call statement for it will simply reduce the code into one for laminar flow calculation. The vorticity evaluation is involved here. Other formulations are provided in section 2.3.

```
      SUBROUTINE BLT3D
      INCLUDE "comm.f"
      COMMON /STATE/ RHO(IM,JM,KM),U(IM,JM,KM),V(IM,JM,KM),W(IM,JM,KM)
```

```
     &           ,T(IM,JM,KM),XMU(IM,JM,KM)
        DIMENSION XMUTI(JM)
C
C --- Baldwin & Lomax Turbulence Model ----------
C --- Calculating wall stress and vorticity ------
C
        DO 15 K=2,KN
        DO 15 I=2,IN
        ASXI1=0.5*(SXI(I,2,K,1)+SXI(I-1,2,K,1))
        ASXI2=0.5*(SXI(I,2,K,2)+SXI(I-1,2,K,2))
        ASXI3=0.5*(SXI(I,2,K,3)+SXI(I-1,2,K,3))
        REXI=ASXI1**2+ASXI2**2+ASXI3**2
        AZET1=0.5*(SZET(I,2,K,1)+SZET(I,2,K-1,1))
        AZET2=0.5*(SZET(I,2,K,2)+SZET(I,2,K-1,2))
        AZET3=0.5*(SZET(I,2,K,3)+SZET(I,2,K-1,3))
        REET=AZET1**2+AZET2**2+AZET3**2
C
        VBJ=SQRT((U(I,2,K)*ASXI1+V(I,2,K)*ASXI2+W(I,2,K)*ASXI3)**2/REXI
     & +(U(I,2,K)*AZET1+V(I,2,K)*AZET2+W(I,2,K)*AZET3)**2/REET)
        DVDN=VBJ/DS(I,2,K)
        RHOW=RHO(I,2,K)
        XMUW=XMU(I,2,K)
        GMAX=0.0
        DO 5 J=2,JN
        UTOP=0.5*(U(I,J,K)+U(I,J+1,K))
        VTOP=0.5*(V(I,J,K)+V(I,J+1,K))
        WTOP=0.5*(W(I,J,K)+W(I,J+1,K))
        IF(J.EQ.2)THEN
        UBOT=0.0
        VBOT=0.0
        WBOT=0.0
        ELSE
        UBOT=0.5*(U(I,J,K)+U(I,J-1,K))
        VBOT=0.5*(V(I,J,K)+V(I,J-1,K))
        WBOT=0.5*(W(I,J,K)+W(I,J-1,K))
        ENDIF
        URIG=0.5*(U(I,J,K)+U(I+1,J,K))
        VRIG=0.5*(V(I,J,K)+V(I+1,J,K))
        WRIG=0.5*(W(I,J,K)+W(I+1,J,K))
        ULEF=0.5*(U(I,J,K)+U(I-1,J,K))
        VLEF=0.5*(V(I,J,K)+V(I-1,J,K))
        WLEF=0.5*(W(I,J,K)+W(I-1,J,K))
        UFOR=0.5*(U(I,J,K)+U(I,J,K+1))
        VFOR=0.5*(V(I,J,K)+V(I,J,K+1))
        WFOR=0.5*(W(I,J,K)+W(I,J,K+1))
        UBAC=0.5*(U(I,J,K)+U(I,J,K-1))
        VBAC=0.5*(V(I,J,K)+V(I,J,K-1))
        WBAC=0.5*(W(I,J,K)+W(I,J,K-1))
C
        DUDY=(URIG*SXI(I,J,K,2)-ULEF*SXI(I-1,J,K,2)
     &    +UTOP*SETA(I,J,K,2)-UBOT*SETA(I,J-1,K,2)
     &    +UFOR*SZET(I,J,K,2)-UBAC*SZET(I,J,K-1,2))
        DVDX=(VRIG*SXI(I,J,K,1)-VLEF*SXI(I-1,J,K,1)
     &    +VTOP*SETA(I,J,K,1)-VBOT*SETA(I,J-1,K,1)
     &    +VFOR*SZET(I,J,K,1)-VBAC*SZET(I,J,K-1,1))
        DVDZ=(VRIG*SXI(I,J,K,3)-VLEF*SXI(I-1,J,K,3)
     &    +VTOP*SETA(I,J,K,3)-VBOT*SETA(I,J-1,K,3)
     &    +VFOR*SZET(I,J,K,3)-VBAC*SZET(I,J,K-1,3))
```

```
        DWDY=(WRIG*SXI(I,J,K,2)-WLEF*SXI(I-1,J,K,2)
     &    +WTOP*SETA(I,J,K,2)-WBOT*SETA(I,J-1,K,2)
     &    +WFOR*SZET(I,J,K,2)-WBAC*SZET(I,J,K-1,2))
        DWDX=(WRIG*SXI(I,J,K,1)-WLEF*SXI(I-1,J,K,1)
     &    +WTOP*SETA(I,J,K,1)-WBOT*SETA(I,J-1,K,1)
     &    +WFOR*SZET(I,J,K,1)-WBAC*SZET(I,J,K-1,1))
        DUDZ=(URIG*SXI(I,J,K,3)-ULEF*SXI(I-1,J,K,3)
     &    +UTOP*SETA(I,J,K,3)-UBOT*SETA(I,J-1,K,3)
     &    +UFOR*SZET(I,J,K,3)-UBAC*SZET(I,J,K-1,3))
C
C -- VORTICITY ----
C
        REVV=1./VOL(I,J,K)
        WW=SQRT((DUDY-DVDX)**2+(DVDZ-DWDY)**2+(DWDX-DUDZ)**2)*REVV
C
        YPLUS=SQRT(RE*RHOW*DVDN/XMUW)*DS(I,J,K)
        GAMA=DS(I,J,K)*(1.-EXP(-YPLUS/26.))
C
        XMUTI(J)=RHO(I,J,K)*RE*(0.41*GAMA)**2*WW
C
        IF(GAMA*WW.GT.GMAX)THEN
        GMAX=GAMA*WW
        YMAX=DS(I,J,K)
        ENDIF
5       CONTINUE
        DO 10 J=2,JN
        FKLEB=1./(1.+5.5*(0.3*DS(I,J,K)/YMAX)**6)
        GWAKE=YMAX*GMAX
        GWAK1=0.25*YMAX*UDIF**2/GMAX
        IF(GWAKE.GT.GWAK1)GWAKE=GWAK1
        XMUTO=0.0168*1.6*RHO(I,J,K)*RE*GWAKE*FKLEB
        IF(XMUTI(J).LT.XMUTO)THEN
        XMUT(I,J,K)=XMUTI(J)
        ELSE
        XMUT(I,J,K)=XMUTO
        ENDIF
10      CONTINUE
C
        XMUT(I,1,K)=XMUT(I,2,K)
15      CONTINUE
        RETURN
        END
```

4.10 Subroutine OUT3D

This subroutine outputs the primitive variables ρ, u, v, w, p, T and Mach number of a station. The array VOL is temporary used here for Mach number. The shifting process for the next station is done here as well.

```
        SUBROUTINE OUT3D
        INCLUDE "comm.f"
        COMMON /STATE/ RHO(IM,JM,KM),U(IM,JM,KM),V(IM,JM,KM),W(IM,JM,KM)
       &              ,T(IM,JM,KM),XMU(IM,JM,KM)
C
        OPEN(1,FILE='rho1.bin',FORM='UNFORMATTED',STATUS='UNKNOWN')
        WRITE(1) (((RHO(I,J,K),I=1,IN+1),J=1,JN+1),K=1,KN+1)
        CLOSE(1)
        OPEN(2,FILE='u1.bin',FORM='UNFORMATTED',STATUS='UNKNOWN')
```

```
      WRITE(2) (((U(I,J,K),I=1,IN+1),J=1,JN+1),K=1,KN+1)
      CLOSE(2)
      OPEN(3,FILE='v1.bin',FORM='UNFORMATTED',STATUS='UNKNOWN')
      WRITE(3) (((V(I,J,K),I=1,IN+1),J=1,JN+1),K=1,KN+1)
      CLOSE(3)
      OPEN(4,FILE='w1.bin',FORM='UNFORMATTED',STATUS='UNKNOWN')
      WRITE(4) (((W(I,J,K),I=1,IN+1),J=1,JN+1),K=1,KN+1)
      CLOSE(4)
      OPEN(1,FILE='t.bin',FORM='UNFORMATTED',STATUS='UNKNOWN')
      WRITE(1) (((T(I,J,K),I=1,IN+1),J=1,JN+1),K=1,KN+1)
      CLOSE(1)
      DO 100 K=1,KN+1
      DO 100 J=1,JN+1
      DO 100 I=1,IN+1
      VT=SQRT(U(I,J,K)**2+V(I,J,K)**2+W(I,J,K)**2)
      CS=SQRT(GAM*T(I,J,K)*GAMC6)
      XMU(I,J,K)=VT/CS
100   CONTINUE
      OPEN(1,FILE='mach.bin',FORM='UNFORMATTED',STATUS='UNKNOWN')
      WRITE(1) (((XMU(I,J,K),I=1,IN+1),J=1,JN+1),K=1,KN+1)
      CLOSE(1)
      DO 200 K=1,KN+1
      DO 200 J=1,JN+1
      DO 200 I=1,IN+1
      XMU(I,J,K)=T(I,J,K)*GAMC6*RHO(I,J,K)
200   CONTINUE
      OPEN(1,FILE='p1.bin',FORM='UNFORMATTED',STATUS='UNKNOWN')
      WRITE(1) (((XMU(I,J,K),I=1,IN+1),J=1,JN+1),K=1,KN+1)
      CLOSE(1)
C Recover XMU
      DO 300 K=1,KN+1
      DO 300 J=1,JN+1
      DO 300 I=1,IN+1
      IF(T(I,J,K).LE.RTINFA)THEN
       XMU(I,J,K)=T(I,J,K)
      ELSE
      XMU(I,J,K)=T(I,J,K)*230.0*SQRT(T(I,J,K)*TINFA)
     & /(T(I,J,K)*TINF+110.0)
      ENDIF
C
300   CONTINUE
      RETURN
      END
```

4.11 Subroutine FLUX

The following two subroutines are the key lower level modules for the inviscid flux evaluation. Arguements C1, C2, C3 are three components of the unit vector of area, AS is the area itself, CS is the local sound speed, UL is the local velocity along the unit vector direction, QV is the column hyper-vector of the primitive variables (ρ, u, v, w, T), Z is the last Riemann invariant and EIB returns the column hyper-vector of the flux. Since we need the flow field information in the global Cartesian coordinate system (x, y, z), while the Riemann invariants must be used in terms of the local coordinate system (ξ, η, ζ), coordinate transformations have to be used twice in subroutine FLUX which is called to evaluate the fluxes other than $\vec{F}_i(Q^L)$

and $\vec{F}_i(Q^R)$. The later two fluxes must be evaluated by calling subroutine FLUX1 in which the calculations are very straightforward.

```
      SUBROUTINE  FLUX(C1,C2,C3,AS,CS,UL,QV,Z,EIB)
      COMMON /GAMS/GAMC0,GAMC1,GAMC2,GAMC3,GAMC4,GAMC5,GAMC6,C0,CT
      DIMENSION QV(5),EIB(5)
      PORHO=CS*CS*GAMC3
      RHO=EXP((ALOG(PORHO)-Z)*GAMC4)
      P=PORHO*RHO
C
      IF(C1.GT.0.2.OR.C1.LT.-0.2)THEN
       VL=C1*QV(3)-C2*QV(2)
       WL=C1*QV(4)-C3*QV(2)
       RVC1=1./C1
       U=UL*C1-VL*C2-WL*C3
       V=(U*C2+VL)*RVC1
       W=(U*C3+WL)*RVC1
      ELSEIF(C2.GT.0.2.OR.C2.LT.-0.2)THEN
       VL=C2*QV(2)-C1*QV(3)
       WL=C2*QV(4)-C3*QV(3)
       RVC2=1./C2
       V=UL*C2-VL*C1-WL*C3
       U=(V*C1+VL)*RVC2
       W=(V*C3+WL)*RVC2
      ELSE
       VL=C3*QV(3)-C2*QV(4)
       WL=C3*QV(2)-C1*QV(4)
       RVC3=1./C3
       W=UL*C3-VL*C2-WL*C1
       U=(W*C1+WL)*RVC3
       V=(W*C2+VL)*RVC3
      ENDIF
C
      VDK=UL*AS
      P1=P*AS
      VSO2=0.5*(U*U+V*V+W*W)
      ET=P*GAMC2+RHO*VSO2
      EIB(1)=RHO*VDK
      EIB(2)=EIB(1)*U+P1*C1
      EIB(3)=EIB(1)*V+P1*C2
      EIB(4)=EIB(1)*W+P1*C3
      EIB(5)=ET*VDK
      RETURN
      END
C
      SUBROUTINE FLUX1(C1,C2,C3,AS,RHO,UL,QV,P,EIB)
      COMMON /GAMS/GAMC0,GAMC1,GAMC2,GAMC3,GAMC4,GAMC5,GAMC6,C0,CT
      DIMENSION QV(5),EIB(5)
C
      U=QV(2)
      V=QV(3)
      W=QV(4)
C
      VDK=UL*AS
      P1=P*AS
      VSO2=0.5*(U*U+V*V+W*W)
```

```
      ET=P*GAMC2+RHO*VSO2
      EIB(1)=RHO*VDK
      EIB(2)=EIB(1)*U+P1*C1
      EIB(3)=EIB(1)*V+P1*C2
      EIB(4)=EIB(1)*W+P1*C3
      EIB(5)=ET*VDK
      RETURN
      END
```

4.12 File Osher.f

This file is designed to be shared by both NS3DT and TPNS3D. In fact, it can be shared by some other programs for three dimensional compressible flows, if the entry and return variables of this module are correctly joined with the module called by. This module could have been made to be a subroutine for the readability of the code, but the efficiency of the code would then be reduced quite much, as the calling of a subroutine of the same function in the three-layer-loop appears obviously slower than the paragraph of the statements inserted in. The sixteen cases of the P-variant Osher's approximate solver are clearly corresponding to the table given in section 3.2.

```
C
C   Osher's Riemann solver (P-variant)
C
C ---- VELOCITY COMPONENTS IN THE DIRECTION NORMAL TO THE SURFACE -----
      UL=C1*QL(2)+C2*QL(3)+C3*QL(4)
      UR=C1*QR(2)+C2*QR(3)+C3*QR(4)
C
      RHOL=QL(1)
      PL=QL(5)*RHOL*GAMC6
      CSL=SQRT(GAM*QL(5)*GAMC6)
C
      RHOR=QR(1)
      PRR=QR(5)*RHOR*GAMC6
      CSR=SQRT(GAM*PQR(5)*GAMC6)
C
C---INTERMEDIATEPOINTS
C
      PHL=UL+GAMC0*CSL
      PHR=UR-GAMC0*CSR
      ZL=ALOG(PL/RHOL**GAM)
      ZR=ALOG(PRR/RHOR**GAM)
      ALFA=EXP((ZR-ZL)*GAMC1)
      C13=(PHL-PHR)/(1.0+ALFA)*0.5*(GAM-1.0)
      C23=ALFA*C13
      UH=(PHR+ALFA*PHL)/(1.0+ALFA)
C
C   ---  THE FOLLOWING PARAMETERS ARE OF NO USE BUT HAVE THEIR MEANINGS.
C
C   ---  Z13=ZL; Z23=ZR; U13=UH; U23=UH; V13=VL; V23=VR; W13=WL; W23=WR
C
      FIB(1)=0.0
      FIB(2)=0.0
      FIB(3)=0.0
      FIB(4)=0.0
      FIB(5)=0.0
      IF(UL.LT.CSL.AND.UR.GE.-CSR)THEN
      IF(UH.GT.C13)THEN
```

```
      US=GAMC5*PHL
      CALL FLUX(C1,C2,C3,AS,US,US,QL,ZL,FIB)
      ELSEIF(UH.GE.0.)THEN
      CALL FLUX(C1,C2,C3,AS,C13,UH,QL,ZL,FIB)
      ELSEIF(UH.GE.-C23)THEN
      CALL FLUX(C1,C2,C3,AS,C23,UH,QR,ZR,FIB)
      ELSE
      US=GAMC5*PHR
      CALL FLUX(C1,C2,C3,AS,US,US,QR,ZR,FIB)
      ENDIF
C
      ELSEIF(UL.GE.CSL.AND.UR.GE.-CSR)THEN
      CALL FLUX1(C1,C2,C3,AS,RHOL,UL,QL,PL,FIB)
      IF(UH.LE.C13)THEN
      US=GAMC5*PHL
      CALL FLUX(C1,C2,C3,AS,US,US,QL,ZL,EIB)
      FIB(1)=FIB(1)-EIB(1)
      FIB(2)=FIB(2)-EIB(2)
      FIB(3)=FIB(3)-EIB(3)
      FIB(4)=FIB(4)-EIB(4)
      FIB(5)=FIB(5)-EIB(5)
      IF(UH.GE.0.0)THEN
      CALL FLUX(C1,C2,C3,AS,C13,UH,QL,ZL,EIB)
      FIB(1)=FIB(1)+EIB(1)
      FIB(2)=FIB(2)+EIB(2)
      FIB(3)=FIB(3)+EIB(3)
      FIB(4)=FIB(4)+EIB(4)
      FIB(5)=FIB(5)+EIB(5)
      ELSEIF(UH.GE.-C23)THEN
      CALL FLUX(C1,C2,C3,AS,C23,UH,QR,ZR,EIB)
      FIB(1)=FIB(1)+EIB(1)
      FIB(2)=FIB(2)+EIB(2)
      FIB(3)=FIB(3)+EIB(3)
      FIB(4)=FIB(4)+EIB(4)
      FIB(5)=FIB(5)+EIB(5)
      ELSE
      US=GAMC5*PHR
      CALL FLUX(C1,C2,C3,AS,US,US,QR,ZR,EIB)
      FIB(1)=FIB(1)+EIB(1)
      FIB(2)=FIB(2)+EIB(2)
      FIB(3)=FIB(3)+EIB(3)
      FIB(4)=FIB(4)+EIB(4)
      FIB(5)=FIB(5)+EIB(5)
      ENDIF
      ENDIF
C
      ELSEIF(UL.LT.CSL.AND.UR.LT.-CSR)THEN
      CALL FLUX1(C1,C2,C3,AS,RHOR,UR,QR,PRR,FIB)
      IF(UH.GE.-C23)THEN
      US=GAMC5*PHR
      CALL FLUX(C1,C2,C3,AS,US,US,QR,ZR,EIB)
      FIB(1)=FIB(1)-EIB(1)
      FIB(2)=FIB(2)-EIB(2)
      FIB(3)=FIB(3)-EIB(3)
      FIB(4)=FIB(4)-EIB(4)
      FIB(5)=FIB(5)-EIB(5)
      IF(UH.LT.0.0)THEN
      CALL FLUX(C1,C2,C3,AS,C23,UH,QR,ZR,EIB)
```

```
      FIB(1)=FIB(1)+EIB(1)
      FIB(2)=FIB(2)+EIB(2)
      FIB(3)=FIB(3)+EIB(3)
      FIB(4)=FIB(4)+EIB(4)
      FIB(5)=FIB(5)+EIB(5)
      ELSEIF(UH.LT.C13)THEN
      CALL FLUX(C1,C2,C3,AS,C13,UH,QL,ZL,EIB)
      FIB(1)=FIB(1)+EIB(1)
      FIB(2)=FIB(2)+EIB(2)
      FIB(3)=FIB(3)+EIB(3)
      FIB(4)=FIB(4)+EIB(4)
      FIB(5)=FIB(5)+EIB(5)
      ELSE
      US=GAMC5*PHL
      CALL FLUX(C1,C2,C3,AS,US,US,QL,ZL,EIB)
      FIB(1)=FIB(1)+EIB(1)
      FIB(2)=FIB(2)+EIB(2)
      FIB(3)=FIB(3)+EIB(3)
      FIB(4)=FIB(4)+EIB(4)
      FIB(5)=FIB(5)+EIB(5)
      ENDIF
      ENDIF
C
      ELSEIF(UL.GE.CSL.AND.UR.LT.-CSR)THEN
      CALL FLUX1(C1,C2,C3,AS,RHOL,UL,QL,PL,FIB)
      CALL FLUX1(C1,C2,C3,AS,RHOR,UR,QR,PRR,EIB)
      FIB(1)=FIB(1)+EIB(1)
      FIB(2)=FIB(2)+EIB(2)
      FIB(3)=FIB(3)+EIB(3)
      FIB(4)=FIB(4)+EIB(4)
      FIB(5)=FIB(5)+EIB(5)
      IF(UH.LT.C13)THEN
      US=GAMC5*PHL
      CALL FLUX(C1,C2,C3,AS,US,US,QL,ZL,EIB)
      FIB(1)=FIB(1)-EIB(1)
      FIB(2)=FIB(2)-EIB(2)
      FIB(3)=FIB(3)-EIB(3)
      FIB(4)=FIB(4)-EIB(4)
      FIB(5)=FIB(5)-EIB(5)
      IF(UH.GE.0.0)THEN
      CALL FLUX(C1,C2,C3,AS,C13,UH,QL,ZL,EIB)
      FIB(1)=FIB(1)+EIB(1)
      FIB(2)=FIB(2)+EIB(2)
      FIB(3)=FIB(3)+EIB(3)
      FIB(4)=FIB(4)+EIB(4)
      FIB(5)=FIB(5)+EIB(5)
      ENDIF
      ENDIF
      IF(UH.GE.-C23)THEN
      US=GAMC5*PHR
      CALL FLUX(C1,C2,C3,AS,US,US,QR,ZR,EIB)
      FIB(1)=FIB(1)-EIB(1)
      FIB(2)=FIB(2)-EIB(2)
      FIB(3)=FIB(3)-EIB(3)
      FIB(4)=FIB(4)-EIB(4)
      FIB(5)=FIB(5)-EIB(5)
      IF(UH.LT.0.0)THEN
      CALL FLUX(C1,C2,C3,AS,C23,UH,QR,ZR,EIB)
```

```
      FIB(1)=FIB(1)+EIB(1)
      FIB(2)=FIB(2)+EIB(2)
      FIB(3)=FIB(3)+EIB(3)
      FIB(4)=FIB(4)+EIB(4)
      FIB(5)=FIB(5)+EIB(5)
      ENDIF
      ENDIF
      ENDIF
```

4.13 Subroutine HTC3D

This subroutine is added for calculation of the heat transfer around the wall. The heat transfer around the wall is considered as a more sensitive parameter than the primitive variables. Once the validation of the code is achieved, the heat transfer around the wall can then be the further check of the accuracy of the code.

The heat transfer rate around the wall can be defined as

$$q = k\left(\frac{\partial T}{\partial n}\right)_{wall} \tag{4.13.1}$$

Because the grid has chosen to be perpendicular to the wall, the calculation of the partial derivative is simple. Using a two point Taylor;s series expansion, one can get

$$\left(\frac{\partial T}{\partial n}\right)_{wall} \approx \frac{\Delta_2^2\left[T_{w+1} - T_w\right] + \Delta_1^2\left[T_w - T_{w+2}\right]}{\Delta_1\Delta_2\left(\Delta_2 - \Delta_1\right)} \tag{4.13.2}$$

where w+1 and w+2 are the two neighbouring points in the outward of wall direction and Δ_1 and Δ_2 are the distances of these points from the wall.

```
      SUBROUTINE HTC3D
      INCLUDE "comm.f"
      DIMENSION QW(IM,KM)
C
C ---- Heat Transfer on the Wall
C
      OPEN(8,FILE='htc.bin',FORM='UNFORMATTED',STATUS='UNKNOWN')
      CP=1004.0
      XMUINF=0.1458E-5*TINF**1.5/(TINF+110.4)
      DO I=2,IN
      DO K=2,KN
      DS1=(DS(I,3,K)+DS(I,2,K))/(2.0*DS(I,2,K))
      DS2=DS(I,3,K)-DS(I,2,K)
      DT1=(T(I,3,K)+T(I,2,K))*0.5
      DT2=(T(I,4,K)+T(I,3,K))*0.5
      QW(I,K)=-CP*XMU(I,1,K)*XMUINF*(DS1*(DT1-TW)+(TW-DT2)/DS1)/DS2*TINF/PR
      END DO
      WRITE(8) (QW(I,K),K=2,KN)
      END DO
      RETURN
      END
```

4.14 Subroutine JKT3D

This subroutine is for the Johnson-King turbulence model. The model has not been tested because it is for flows with strong upstream separation, and we have not found a suitable test case for it. The INCLUDE statement is for COMMON declaration, and there are some more arrays for storing the variables needed for the calculation. All the necessary comments are stated in the subroutine where some new variables are introduced.

This subroutine must co-operate with subroutine BLT3D and RSS3D which is introduced in the next section.

```
      SUBROUTINE JKT3D
      INCLUDE "commn.f"
      COMMON /STATE/ RHO(IM,JM,KM),U(IM,JM,KM),V(IM,JM,KM),W(IM,JM,KM)
     &             ,T(IM,JM,KM),XMU(IM,JM,KM)
      DIMENSION XMIML(JM),XMIJK(JM)
C
C --- Johnson & King Turbulence Model --- Added in by Dr. Xiaoshi Jin
C     in Oct.,1992.
C
      DO 15 K=2,KN
      DO 15 I=2,IN
      ASXI1=0.5*(SXI(I,2,K,1)+SXI(I-1,2,K,1))
      ASXI2=0.5*(SXI(I,2,K,2)+SXI(I-1,2,K,2))
      ASXI3=0.5*(SXI(I,2,K,3)+SXI(I-1,2,K,3))
      REXI=ASXI1**2+ASXI2**2+ASXI3**2
      AZET1=0.5*(SZET(I,2,K,1)+SZET(I,2,K-1,1))
      AZET2=0.5*(SZET(I,2,K,2)+SZET(I,2,K-1,2))
      AZET3=0.5*(SZET(I,2,K,3)+SZET(I,2,K-1,3))
      REET=AZET1**2+AZET2**2+AZET3**2
C
      VBJ=SQRT((U(I,2,K)*ASXI1+V(I,2,K)*ASXI2+W(I,2,K)*ASXI3)**2/REXI
     & +(U(I,2,K)*AZET1+V(I,2,K)*AZET2+W(I,2,K)*AZET3)**2/REET)*VINF
      DVDN=VBJ/DS(I,2,K)
      RHOW=RHO(I,2,K)
C
C --- Calculating wall shear stress and vorticity ------
C
      XMUW=XMU(I,2,K)
      GMX=0.0
      GMN=10000.0
      GWW=0.0
      TAUM1=0.0
      DO 5 J=2,JN
      UTOP=0.5*(U(I,J,K)+U(I,J+1,K))
      VTOP=0.5*(V(I,J,K)+V(I,J+1,K))
      WTOP=0.5*(W(I,J,K)+W(I,J+1,K))
      IF(J.EQ.2)THEN
      UBOT=0.0
      VBOT=0.0
      WBOT=0.0
      ELSE
      UBOT=0.5*(U(I,J,K)+U(I,J-1,K))
      VBOT=0.5*(V(I,J,K)+V(I,J-1,K))
      WBOT=0.5*(W(I,J,K)+W(I,J-1,K))
      ENDIF
      URIG=0.5*(U(I,J,K)+U(I+1,J,K))
      VRIG=0.5*(V(I,J,K)+V(I+1,J,K))
      WRIG=0.5*(W(I,J,K)+W(I+1,J,K))
      ULEF=0.5*(U(I,J,K)+U(I-1,J,K))
      VLEF=0.5*(V(I,J,K)+V(I-1,J,K))
```

```
          WLEF=0.5*(W(I,J,K)+W(I-1,J,K))
          UFOR=0.5*(U(I,J,K)+U(I,J,K+1))
          VFOR=0.5*(V(I,J,K)+V(I,J,K+1))
          WFOR=0.5*(W(I,J,K)+W(I,J,K+1))
          UBAC=0.5*(U(I,J,K)+U(I,J,K-1))
          VBAC=0.5*(V(I,J,K)+V(I,J,K-1))
          WBAC=0.5*(W(I,J,K)+W(I,J,K-1))
C
          DUDY=URIG*SXI(I,J,K,2)-ULEF*SXI(I-1,J,K,2)
         &   +UTOP*SETA(I,J,K,2)-UBOT*SETA(I,J-1,K,2)
         &   +UFOR*SZET(I,J,K,2)-UBAC*SZET(I,J,K-1,2)
          DVDX=VRIG*SXI(I,J,K,1)-VLEF*SXI(I-1,J,K,1)
         &   +VTOP*SETA(I,J,K,1)-VBOT*SETA(I,J-1,K,1)
         &   +VFOR*SZET(I,J,K,1)-VBAC*SZET(I,J,K-1,1)
          DVDZ=VRIG*SXI(I,J,K,3)-VLEF*SXI(I-1,J,K,3)
         &   +VTOP*SETA(I,J,K,3)-VBOT*SETA(I,J-1,K,3)
         &   +VFOR*SZET(I,J,K,3)-VBAC*SZET(I,J,K-1,3)
          DWDY=WRIG*SXI(I,J,K,2)-WLEF*SXI(I-1,J,K,2)
         &   +WTOP*SETA(I,J,K,2)-WBOT*SETA(I,J-1,K,2)
         &   +WFOR*SZET(I,J,K,2)-WBAC*SZET(I,J,K-1,2)
          DWDX=WRIG*SXI(I,J,K,1)-WLEF*SXI(I-1,J,K,1)
         &   +WTOP*SETA(I,J,K,1)-WBOT*SETA(I,J-1,K,1)
         &   +WFOR*SZET(I,J,K,1)-WBAC*SZET(I,J,K-1,1)
          DUDZ=URIG*SXI(I,J,K,3)-ULEF*SXI(I-1,J,K,3)
         &   +UTOP*SETA(I,J,K,3)-UBOT*SETA(I,J-1,K,3)
         &   +UFOR*SZET(I,J,K,3)-UBAC*SZET(I,J,K-1,3)
C
C -- VORTICITY ----
C
          REVV=1./VOL(I,J,K)
          WW=SQRT((DUDY-DVDX)**2+(DVDZ-DWDY)**2+(DWDX-DUDZ)**2)*REVV
C
          TAU=XMUT(I,J,K)*WW/RHO(I,J,K)
          IF(TAU.GT.TAUM1)TAUM1=TAU
          IF(G(I,J,K).GT.0.0.AND.G(I,J,K).LT.GMN)THEN
          GMN=G(I,J,K)
          JM1(I,K)=J
          ENDIF
C
          YPLUS=SQRT(RE*RHOW*DVDN/XMUW)*DS(I,J,K)
          GAMA=DS(I,J,K)*(1.-EXP(-YPLUS/26.))
C
C  This is Prandtl-Van Driest mixing-length formulation used in B-L model
C
          XMIML(J)=RHO(I,J,K)*RE*(0.41*GAMA)**2*WW
C
          GWY=GAMA*WW
C
C Select the farthest peak away from the wall (at most three peaks).
C
          IF(GWY.GT.GMX)THEN
          GMX=GWY
          YMX=DS(I,J,K)
          ENDIF
 5        CONTINUE
C
C UMM is defined by Johnson & Coakley, p2001, v.28, AIAA J.
C This is the maximum Reynolds shear stress along I,K ray
```

```
C
      UMM=1./G(I,JM1(I,K),K)
C
      UT=AMAX1(UMM/SQRT(RHO(I,JM1(I,K),K)),SQRT(XMUW*DVDN/RHOW))
C
C Ut is given by p1428, v.28 AIAA J. by Abid, et al.
C
C DELTA is the boundary layer thickness
C ELNGM is the dissipation length defined by, p1428, V.28 AIAA Journal
C
      DELTA=1.9*YMAX
      ELNGM=AMIN1(0.4*DS(I,JM1(I,K),K),0.09*DELTA)
C
C SIGMA is the modeling parameter which links the assumed eddy-viscosity
C    distribution and the rate equation for the streamwise development
C    of the maximum Reynolds shear stress
C
      SIGMA(I,K)=SIGMA(I,K)*UMM**2/TAUM1
C
C ELC is Lc given by J-C & in GEE, et al. p658 Vol 30, AIAA Journal
C
      ELC=SQRT(RHOW)*UT*ELNGM/(SQRT(RHOW)*UT+
     &  SQRT(RHO(I,JM1(I,K),K)*UMM))
      GWAKE=YMAX*GMAX
C
      DO 10 J=2,JN
C
      YPLUS=DS(I,J,K)*RHOW*UT*SQRT(RE)/XMUW
C
C DD is the damping factor given by p658, v.30 AIAA J.
C
      DD=1.-EXP(-YPLUS/17.)
C
C This is modified J-K model defined by J-C, p2001, v.28, AIAA J.
C where the eddy velocity scale Us is given as
C
      RL1=DS(I,J,K)/ELC
      GAMA2=TANH(RL1)
      IF(RL1.LE.1.)THEN
      US=SQRT(RHOW/RHO(I,J,K))*(1.0-RL1)*UT
     &   +SQRT(RHO(I,JM1(K),K)/RHO(I,J,K))*RL1*UMM
      ELSE
      US=SQRT(RHOW/RHO(I,J,K))*(1.0-GAMA2)*UT
     &   +SQRT(RHO(I,JM1(K),K)/RHO(I,J,K))*GAMA2*UMM
      ENDIF
      XMIJK=RHO(I,J,K)*0.4*RE*DD**2*DS(I,J,K)*US
C
C  GEQ is determined from the equilibrium Reynolds shear stress, which
C     is solved from the same ODE by assuming GAMMA is zero.
C
      GAMMA=0.25/(2.0*ELNGM)*(1.0-G(I,J,K)/GEQ(I,J,K))
     &      +0.5*ABS(1.-SQRT(SIGMA(I,K)))/
     &      (0.7*DELTA-DS(I,JM1(I,K),K))
C
C Right-hand-side of the rate-equation
C
      RR(I,J,K,1)=-VOL(I,J,K)*GAMMA
C
```

```
C  This is the inner eddy viscosity
C
      XMUTI=(1.-GAMA2)*XMIML(J)+GAMA2*XMIJK
C
C This is the outer eddy viscosity
C
      FKLEB=1./(1.+5.5*(DS(I,J,K)/DELTA)**6)
      XMUTO=0.0168*0.16*RHO(I,J,K)*RE*GWAKE*FKLEB*SIGMA(I,K)
C
C  This is the eddy viscosity after such a complicated calculation
C
      XMUT(I,J,K)=XMUTO*TANH(XMUTI/XMUTO)
10    CONTINUE
      XMUT(I,1,K)=XMUT(I,2,K)
15    CONTINUE
      RETURN
      END
```

4.15 Subroutine RSS3D

This subroutine is for evaluation of the maximum Reynolds shear stress. Euler's method is used for the partial diferential equation of the maximum Reynolds shear stress. Array G(IM,JM,KM) is introduced for the Reynolds shear stress, and RR is used to store the right-hand-side of the equation (see section 2.4), and residual of the integration.

```
      SUBROUTINE RSS3D
      INCLUDE "commn.f"
      COMMON /STATE/ PVR(IM,JM,KM,5),XMU(IM,JM,KM)
      DIMENSION QF(3),QR(3),QU(3)
C
C Euler's method for the PDE of the maximum Reynolds shear stress
C G(IM,JM,KM) is the array for 1/SQRT(TAU), RR(I,J,K,1) is the right-
C hand-side, RR(I,J,K,2) is the residual for the integral.
C
      DO K=1,KN
      DO J=1,JN
      DO I=1,IN
C
      GG=G(I,J,K)
      GF=0.5*(GG+G(I+1,J,K))
      GR=0.5*(GG+G(I,J+1,K))
      GU=0.5*(GG+G(I,J,K+1))
C
      QF(1)=0.5*(PVR(I,JM1(I,K),K,2)+PVR(I+1,JM1(I+1,K),K,2))
      QF(2)=0.5*(PVR(I,JM1(I,K),K,3)+PVR(I+1,JM1(I+1,K),K,3))
      QF(3)=0.5*(PVR(I,JM1(I,K),K,4)+PVR(I+1,JM1(I+1,K),K,4))
      QR(1)=PVR(I,JM1(I,K),K,2)
      QR(2)=PVR(I,JM1(I,K),K,3)
      QR(3)=PVR(I,JM1(I,K),K,4)
      QU(1)=0.5*(PVR(I,JM1(I,K),K,2)+PVR(I,JM1(I,K+1),K+1,2))
      QU(2)=0.5*(PVR(I,JM1(I,K),K,3)+PVR(I,JM1(I,K+1),K+1,3))
      QU(3)=0.5*(PVR(I,JM1(I,K),K,4)+PVR(I,JM1(I,K+1),K+1,4))
C
      DIF=QF(1)*SXI(I,J,K,1)+QF(2)*SXI(I,J,K,2)+QF(3)*SXI(I,J,K,3)
      DIR=QR(1)*SETA(I,J,K,1)+QR(2)*SETA(I,J,K,2)+QR(3)*SETA(I,J,K,3)
      DIU=QU(1)*SZET(I,J,K,1)+QU(2)*SZET(I,J,K,2)+QU(3)*SZET(I,J,K,3)
C
      RR(I,J,K,2)=RR(I,J,K,2)+RR(I,J,K,1)+DIF*GF+DIR*GR+DIU*GU
```

```
          RR(I+1,J,K,2)=RR(I+1,J,K,2)-DIF*GF
          RR(I,J+1,K,2)=RR(I,J+1,K,2)-DIR*GR
          RR(I,J,K+1,2)=RR(I,J,K+1,2)-DIF*GU
C
C  This is for GEQ, the equilibrium Reynolds shear stress, which
C      is solved from the same ODE by assuming GAMMA is zero.
C
          RR(I,J,K,3)=RR(I,J,K,3)+DIF*GF+DIR*GR+DIU*GU
          RR(I+1,J,K,3)=RR(I+1,J,K,3)-DIF*GF
          RR(I,J+1,K,3)=RR(I,J+1,K,3)-DIR*GR
          RR(I,J,K+1,3)=RR(I,J,K+1,3)-DIF*GU
          END DO
          END DO
          END DO
C
          DO K=2,KN
          DO J=2,JN
          DO I=2,IN
C
C-- LOCOL TIME STEP CALCULATION ------------------------------
C
          UU=SXI(I,J,K,1)*PVR(I,J,K,2)
         &  +SXI(I,J,K,2)*PVR(I,J,K,3)
         &  +SXI(I,J,K,3)*PVR(I,J,K,4)
          VV=SETA(I,J,K,1)*PVR(I,J,K,2)
         &  +SETA(I,J,K,2)*PVR(I,J,K,3)
         &  +SETA(I,J,K,3)*PVR(I,J,K,4)
          WW=SZET(I,J,K,1)*PVR(I,J,K,2)
         &  +SZET(I,J,K,2)*PVR(I,J,K,3)
         &  +SZET(I,J,K,3)*PVR(I,J,K,4)
          CS=SQRT(GAM*PVR(I,J,K,5)*GAMC6)
          SA1=AREA(I,J,K,1)
          SA2=AREA(I,J,K,2)
          SA3=AREA(I,J,K,3)
          SOUND=(SA1+SA2+SA3)*CS
          REC1=1.0/(PVR(I,J,K,1)*RE*PR*VOL(I,J,K))
          DTL=CFL/(ABS(UU)+ABS(VV)+ABS(WW)+SOUND+2.0*GAM*REC1*
        & (XMU(I,J,K)+XMUT(I,J,K))*(SA1**2+SA2**2+SA3**2))
          G(I,J,K)=G(I,J,K)-DTL*RR(I,J,K,2)
          IF(G(I,J,K).LT.0.0)G(I,J,K)=-G(I,J,K)
          GEQ(I,J,K)=GEQ(I,J,K)-DTL*RR(I,J,K,3)
          IF(GEQ(I,J,K).LT.0.0)GEQ(I,J,K)=-GEQ(I,J,K)
C
          END DO
          END DO
          END DO
          RETURN
          END
```

4.16 Paralellisation

A parallelised version of this code has also been successfully completed. It has been running on the Intel Supercomputer System iPSC/860 in Daresbury through the network. Each processing node there has 16Mbytes memory and 80 MFlops speed for single precision. The architecture of the system is hypercube configurated so that the number of the processors used must be 2^n where n is up to 5. The basic idea for parallel computing here is to divide the grid into a number of portions which will be running separately in the same number of the

processors, each grid portion has a part to overlap with its adjacent portion, and this part of computed data will be transferred to the node for the adjacent portion of the grid. Since the communication speed can never match the processing speed, the efficiency is then built on the balance between the amount of data for computing and those for communication. A typical subroutine used in NS3DT code for communication in parallel computing is listed below:

```
      SUBROUTINE SEN3D
      INCLUDE "comm.f"
      DIMENSION buff(JM,KM,5)
C
      IF(node.NE.0)THEN
      len=40*JM*KM
      nod0=100+IT+node-1
      call crecv(nod0,buff,len)
      DO 100 K=1,KM
      DO 100 J=1,JM
      DO 100 L=1,5
      RR(is,J,K,L)=RR(is,J,K,L)+buff(J,K,L)
100   CONTINUE
      ENDIF
      IF(node.NE.lnode)THEN
      DO 200 K=1,KM
      DO 200 J=1,JM
      DO 200 L=1,5
      buff(J,K,L)=RR(ie+1,J,K,L)
200   CONTINUE
      len=40*JM*KM
      nod0=100+IT+node
      nod1=node+1
      call csend(nod0,buff,len,nod1,0)
      ENDIF
      RETURN
      END
```

In this subroutine, ***csend*** and ***crecv*** are two communication subroutines supplied in the system. The array ***buff*** is the buffer storage for transferring the overlaid part of the residual array RR. ***is*** and ***ie*** correspond to the starting and ending of the **I** loop for a processor.

Since parallel computing systems may be different, the subroutines for communication are coded in a few "system dependent" modules so that replanting the code could be easily done.

5. USER'S GUIDE

Before starting to run the code, one must prepare two files which contain the control parameters and the grid of the whole problem, respectively. The file for control parameters has a fixed file name called "inp3dt.data", which is designed for batch job use or background run. The grid file name must be supplied as a control parameter in the input file.

5.1 Input file:

The file inp3dt.data must be generated by a text-editor, and with a slight change, it can also be used as the input file for TPNS3D. The data are arranged as such that each figure is described on the top of it, to remind the user of the meanings. An example data file is listed hereafter which is for the ogive cylinder case, and the results of which are to be shown in the next section as the validation test.

[Begin of File]

```
      DRA Ogive
- IORDER -- ITYPE  -- INTVAR -  ISHARP
     3          1          10          0
- ALPHA  --  CFL   -- CRIIT  -- EPS    -- CRIT1
    18.0       0.3      0.1E-7   1.0E-6      1.0
- XMINF  --  TINF  --   TW   --   RE     --   PR   -- GAM
     3.5       91.0      280.0  0.131E7     0.72      1.4
-    IN    --    JN    --    KN   -- ITMAX  -- IRST
     55         65         75        5000        0
ogive.bin                 GRID FILE NAME
[End of File]
```

The first line gives the title of the problem, and/or a description of it with no more than 80 characters. The names of the parameters are given correspondingly on top of each lines except the last two parameters which are followed by their interpretations. Most of the parameters have been explained in the section 4.2, and with these names and interpretations it should not be difficult to understand each figure.

5.2 Grid file

The grid file is generated by a mesh generator, and it must be stored unformatted. The order of the coordinates must be as follows:

```
DO I=1,IN
 ((X(J,K),Y(J,K),Z(J,K),J=1,JN),K=1,KN)
END DO
```

from which it can be seen that the I stations have to be read into the program one by one. The I-increasing direction must coincide with positive z direction, which is also the down-stream flow direction. The I-surfaces are not necessary plane, but the stretching lines from the object in each of the I-surfaces are required to be perpendicular to the wall, and they are better to be straight lines for accurate turbulent modellings. The first J-surface must be the wall surface of the object, while the first and the last K-surfaces are lain on the symmetrical plane. A schematical view of the grid requirement is shown below.

The grids of the following cases are generated by CFDRAW[12]. The module of the mesh generator can also be separated from it for independent use, or attached to the program if it is necessary. The user's guide for the mesh generator can be found in reference [12].

5.3 Starting

There are two ways for starting the calculation: starting or restarting. For starting, only the control input file and the grid file are needed, for restarting, the flow field has to be supplied. The flow field of the physical variables is supplied by following five files:

rho.bin — the file of the density of the starting station;
t.bin — the file of the pressure of the starting station;
u.bin — the file of the x-component of the velocity of the starting station;
v.bin — the file of the y-component of the velocity of the starting station;
w.bin — the file of the z-component of the velocity of the starting station.
These files are written unformattedly in the following order (e.g. temperature):

```
DO I=1,IN
 WRITE(26) ((T(J,K),J=1,JN),K=1,KN)
END DO
```

The flow field of the output in the previous run can be used for restarting.

5.4 Monitoring and Result files

The progress of the computation is recorded in a file called "check.data" in which one can trace the history of convergence. The meanings of the eight columns in the file from left to right are iterarion counter, the second norm of the total residual, the absolute value of the total residual, I,J, K numbers and the component number of the fluxes at which the maximum residual occurs, total CPU.

The computational results are stored in the following files, which in the order of the primitive variables are rho1.bin, u1.bin, v1.bin, w1.bin, t1.bin, p1.bin and mach.bin.

6.VALIDATION OF THE CODE

6.1 Test Case I

The test case is the Ogive Cylinder, the mesh of which is shown in Fig. 6.2.1, and the conditions are given as follow:

Free stream Mach number	3.5
Free stream temperature	91.0
Wall temperature	280.0
Reynolds number	1310000
Angle of Attack	18.0

The test case was run in Daresbury's Intel iPSC/860, and eight processors were used. The maximum numbers of I,J and K are 55,65,75, respectively. Grid portion 2,4,6,8 are shown is shown in Fig. 6.1.1.

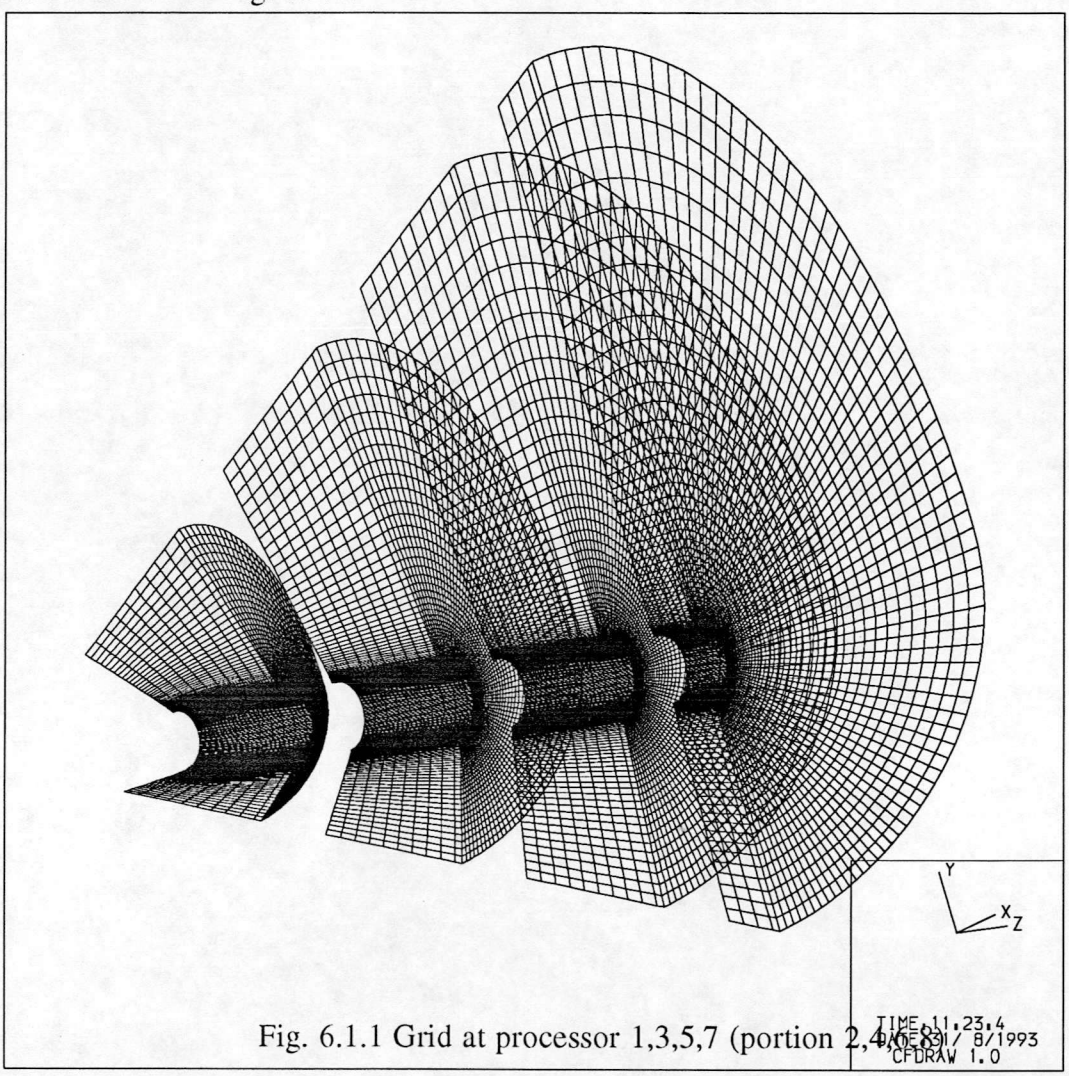


Fig. 6.1.1 Grid at processor 1,3,5,7 (portion 2,4,6,8)

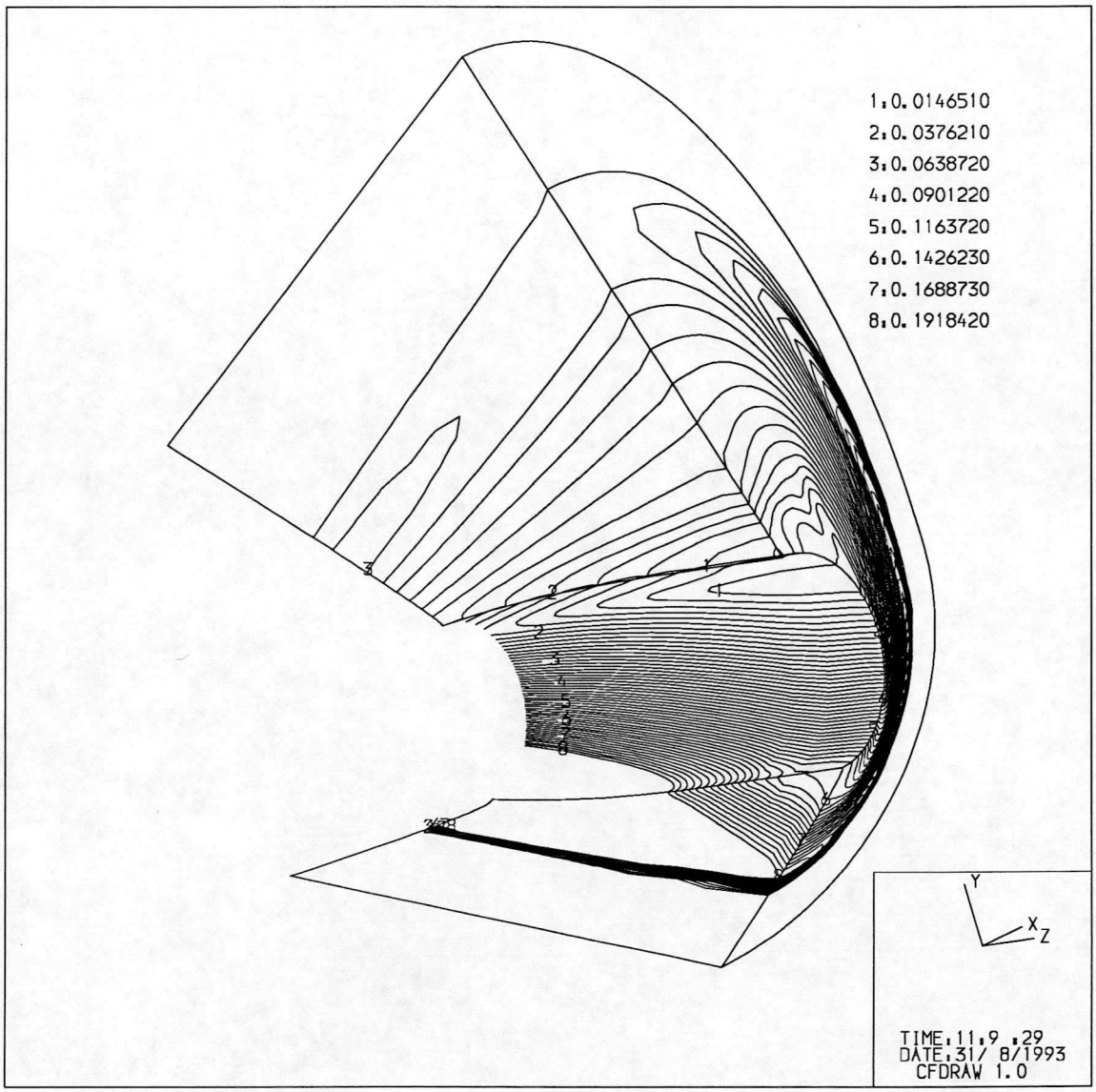


Fig. 6.1.2 The pressure contours of grid portion 2, I=8-18

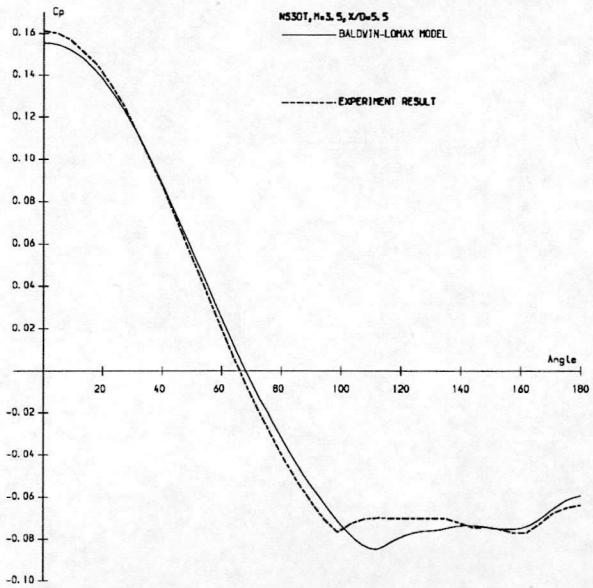


Fig. 6.1.3 Comparison with experimental result: Cp-Angle at x/D=5.5

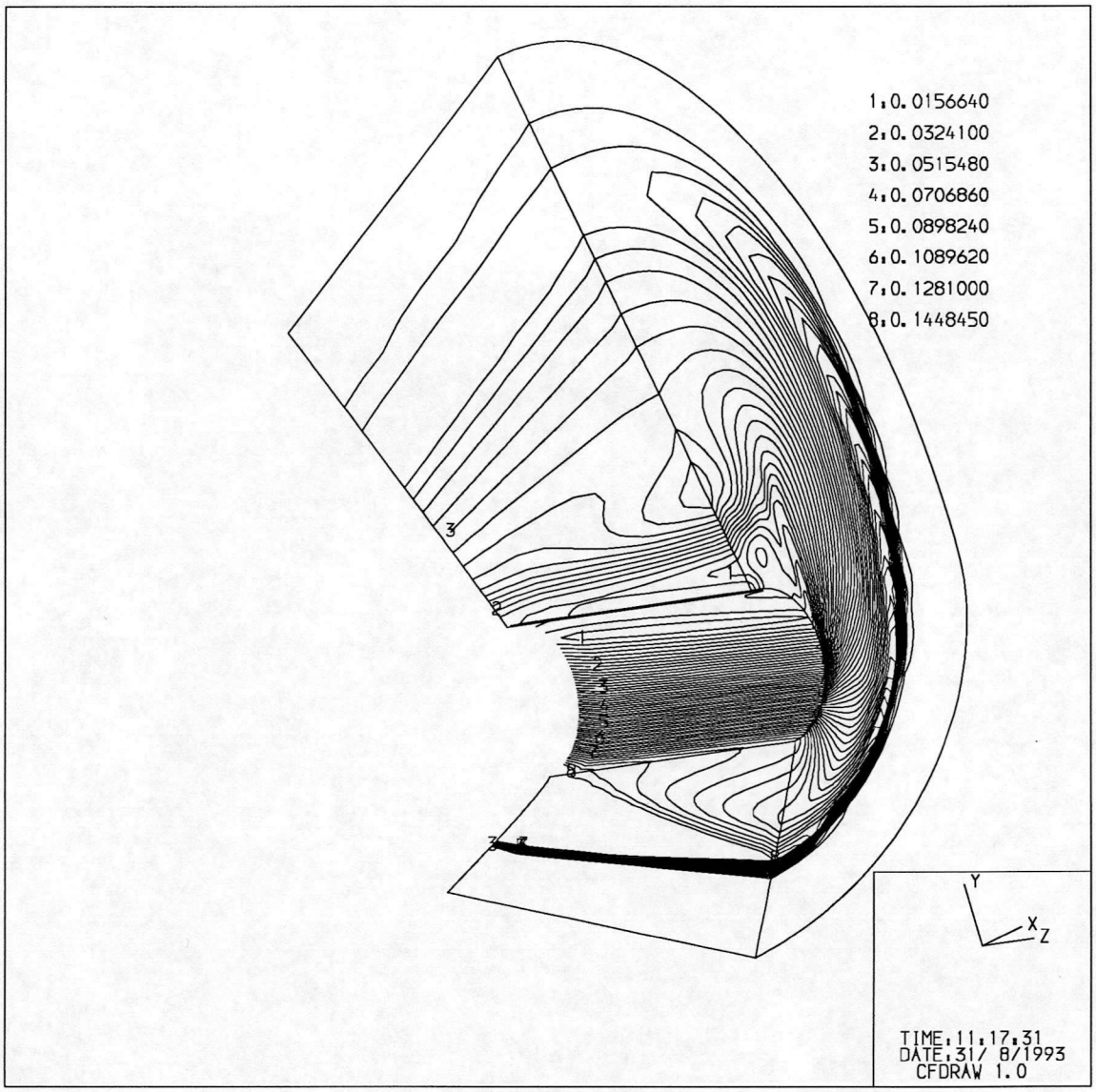


Fig. 6.1.4 The pressure contours of grid portion 3, I=15-25

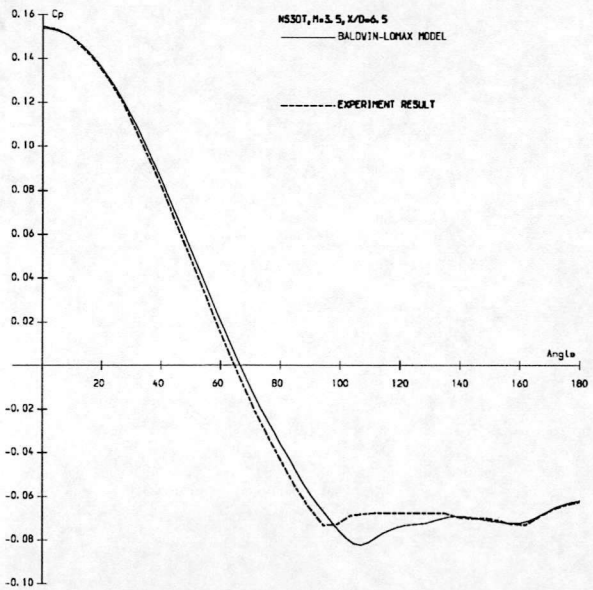


Fig. 6.1.5 Comparison with experimental result: Cp-Angle at x/D=6.5

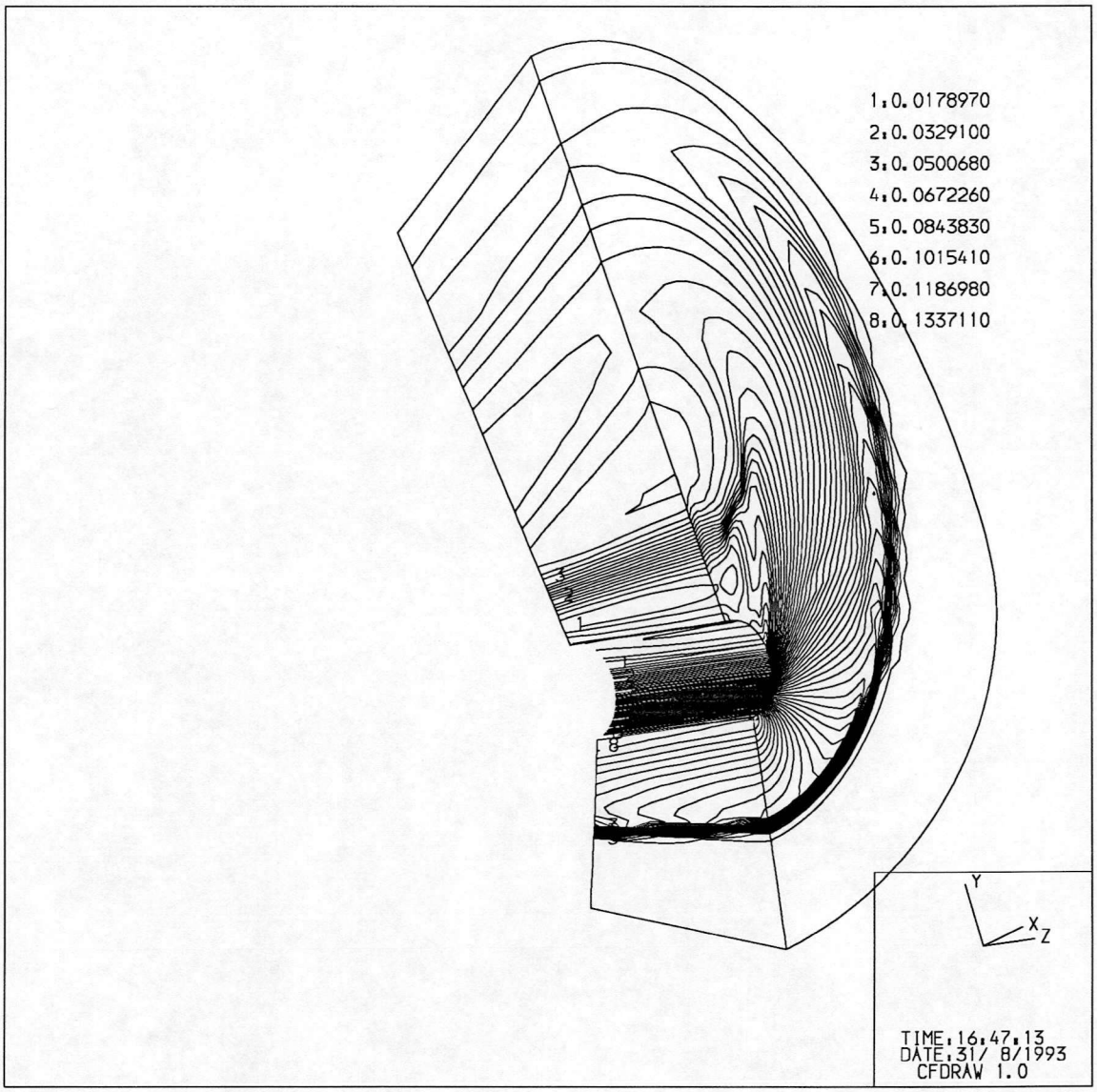


Fig. 6.1.6 The pressure contours of grid portion 5, I=29-39

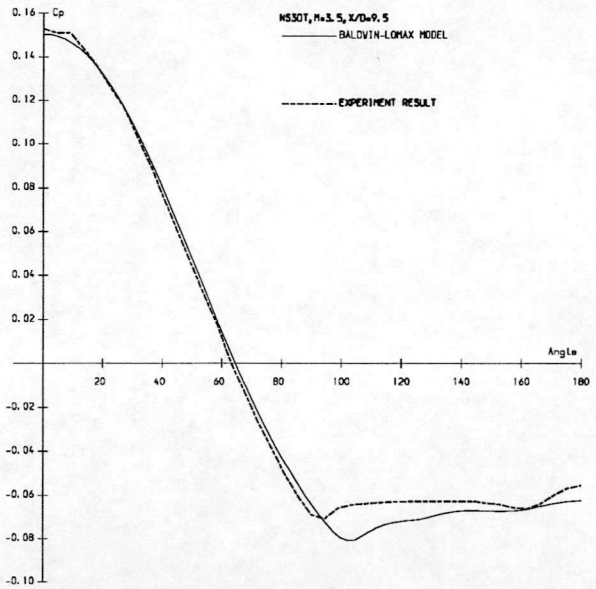


Fig. 6.1.7 Comparison with experimental result: Cp-Angle at x/D=9.5

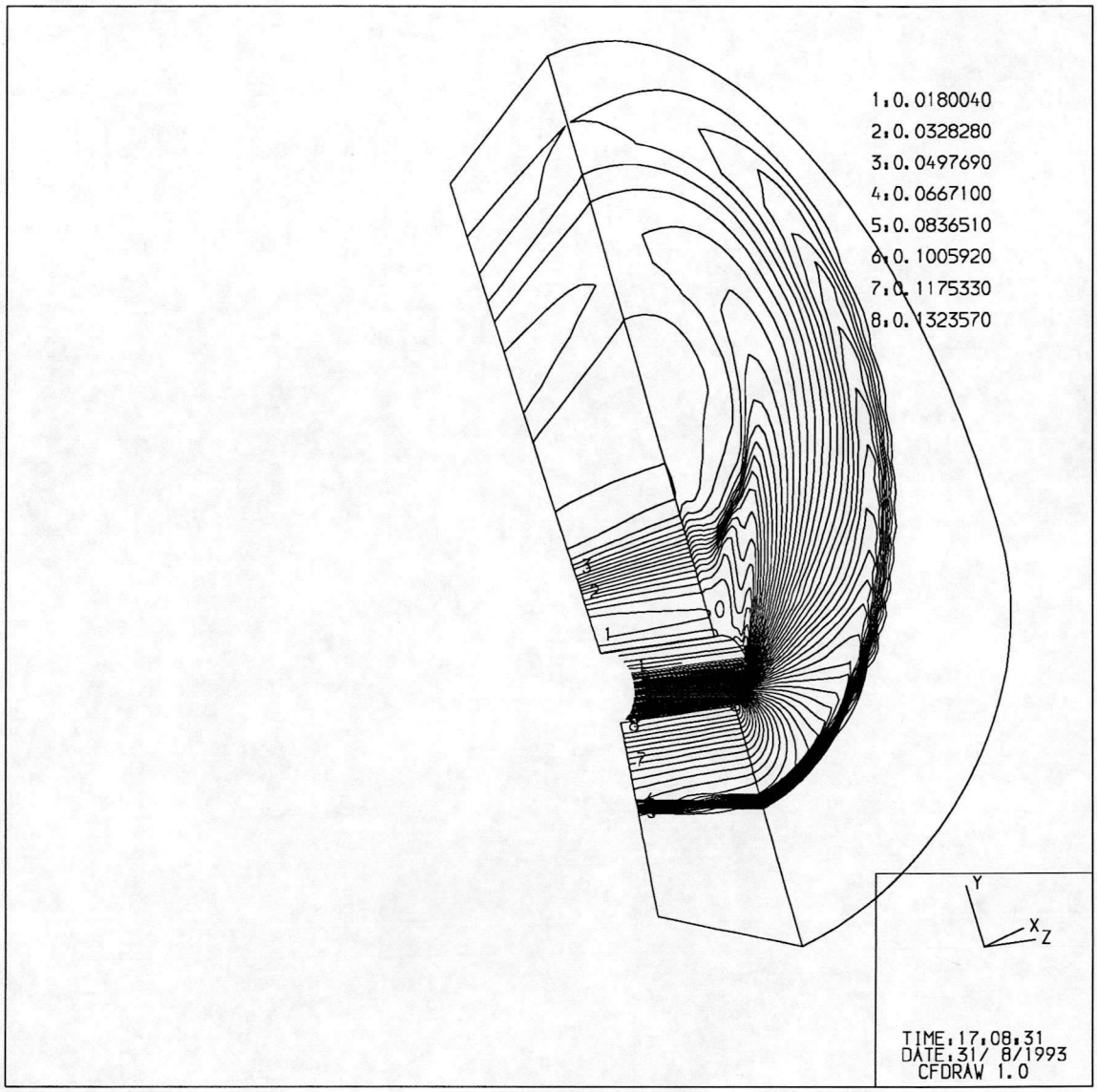


Fig. 6.1.8 The pressure contours of grid portion 7, I=43-53

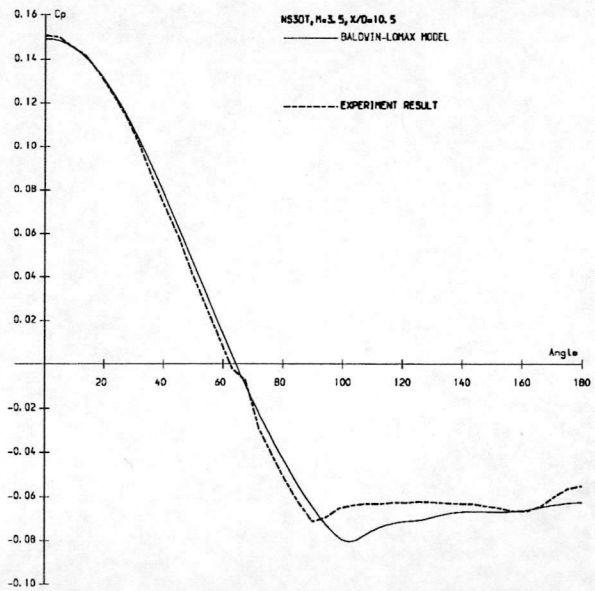


Fig. 6.1.9 Comparison with experimental result: Cp-Angle at x/D=10.5

The same test case has been used for validating TPNS3D code. The grid, however, is not the same as the one used for TPNS3D. Around the nose it gives more curved surfaces, with the first I surface converging into an axis. This kind of grid could result in difficulty for turbulence modelling in evaluating some parameters along a radical ray if there was turbulent flow to appear there. Although there is not much turbulent phenomenon appearing in the fore head area, the transition section from the laminar flow to turbulent flow is a problem. Since we have not started to apply a model for a gradual transferring laminar flow to turbulent flow, a sudden switch from laminar to turbulent flow could result in numerical jump of physical values.

The same problem also appears in the second case, as it can be seen in the validation of TPNS3D[9]. Therefore in this NS3DT code we decide to start the turbulence modelling in the whole calculation domain after a few hundreds iterations of laminar flow calculation. The fore head area thus cannot be expected to have very good agreement with the experimental result, but the rest of the domain should be pretty well predicted. The agreement can be seen in the Cp-Angle plots in figures 6.1.3, 6.1.5, 6.1.7 and 6.1.9 in which the computed results are compared against the measured circumferential surface pressure distributions in four axial locations (x/D = 5.5,6.5,9.5 and 10.5).

Figures 6.2.2, 6.2.4, 6.2.6 and 6.2.8 show the 3D pressure contours in the calculation domain at processor 2, 3, 5 and 7, respectively. These contour plots display the pressure distribution in the cross section view as well as its development along the streamwise direction. The I numbers in portion 2 and portion 3 shown in figures 6.1.2 and 6.1.4, respectively, give the basic idea of the paralellisation of the computation, in which there are three I stations are actually overlaped between proceesors. The amount of the information for communication can be easily estimated. If 16 or more processors were used for the grid it would not increase the speed of computation because the communication would be too much more comparing with the computation within each processor.

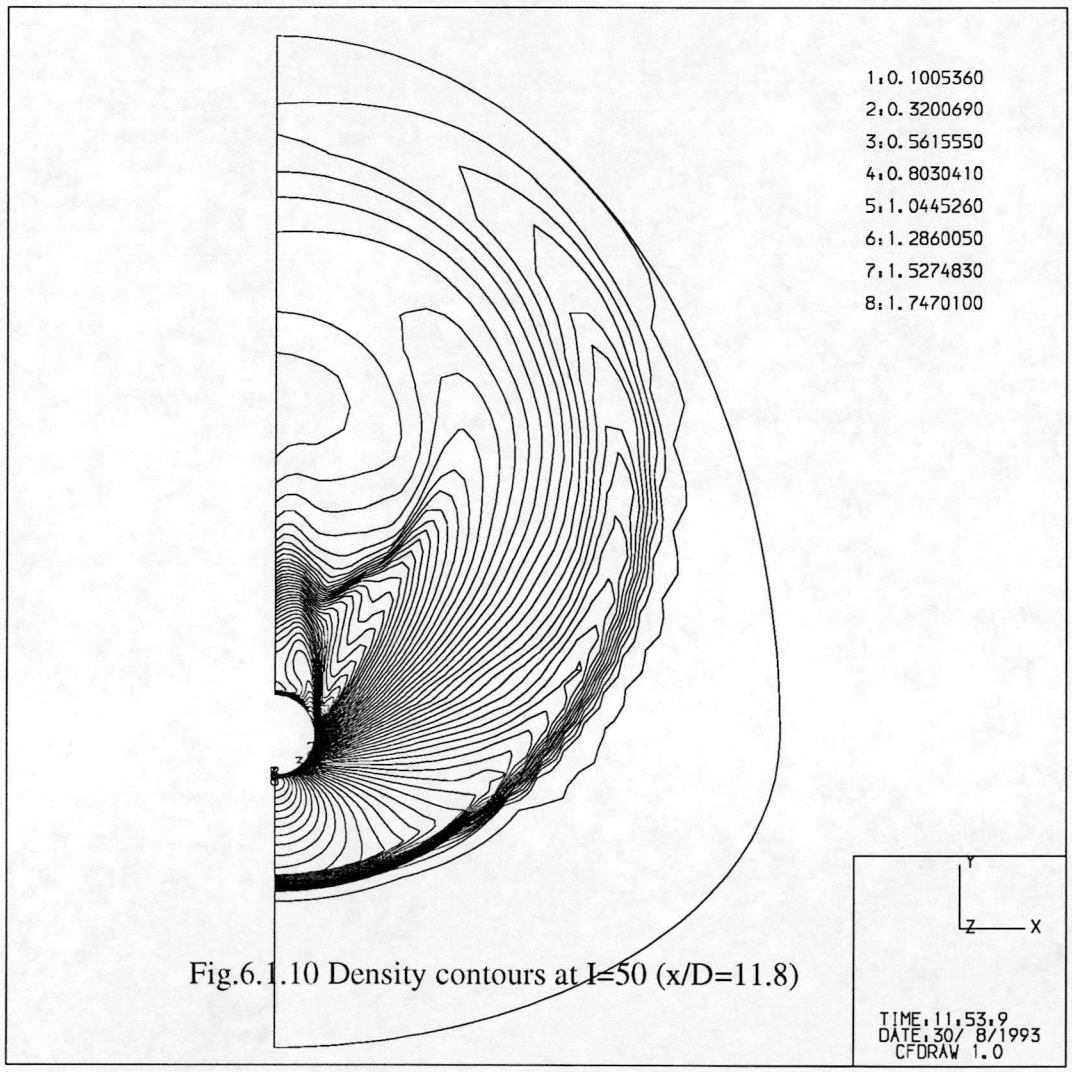


Fig.6.1.10 Density contours at I=50 (x/D=11.8)

6.2 Test Case II

The second test case is also the Ogive Cylinder with different flow conditions which are given as follow:

Free stream Mach number	2.5
Total temperature of free stream	136.89
Wall temperature	288.0
Reynolds number	1219200
Angle of Attack	14.0

The grid of this test case is similar to the one used for the other test case, with the maximum numbers of I, J and K are 55,65,75, respectively. The grid of the fore body (the first twenty I stations) is shown in Fig.6.2.1. The same case with the same grid is also used for validating TPNS3D code[9].

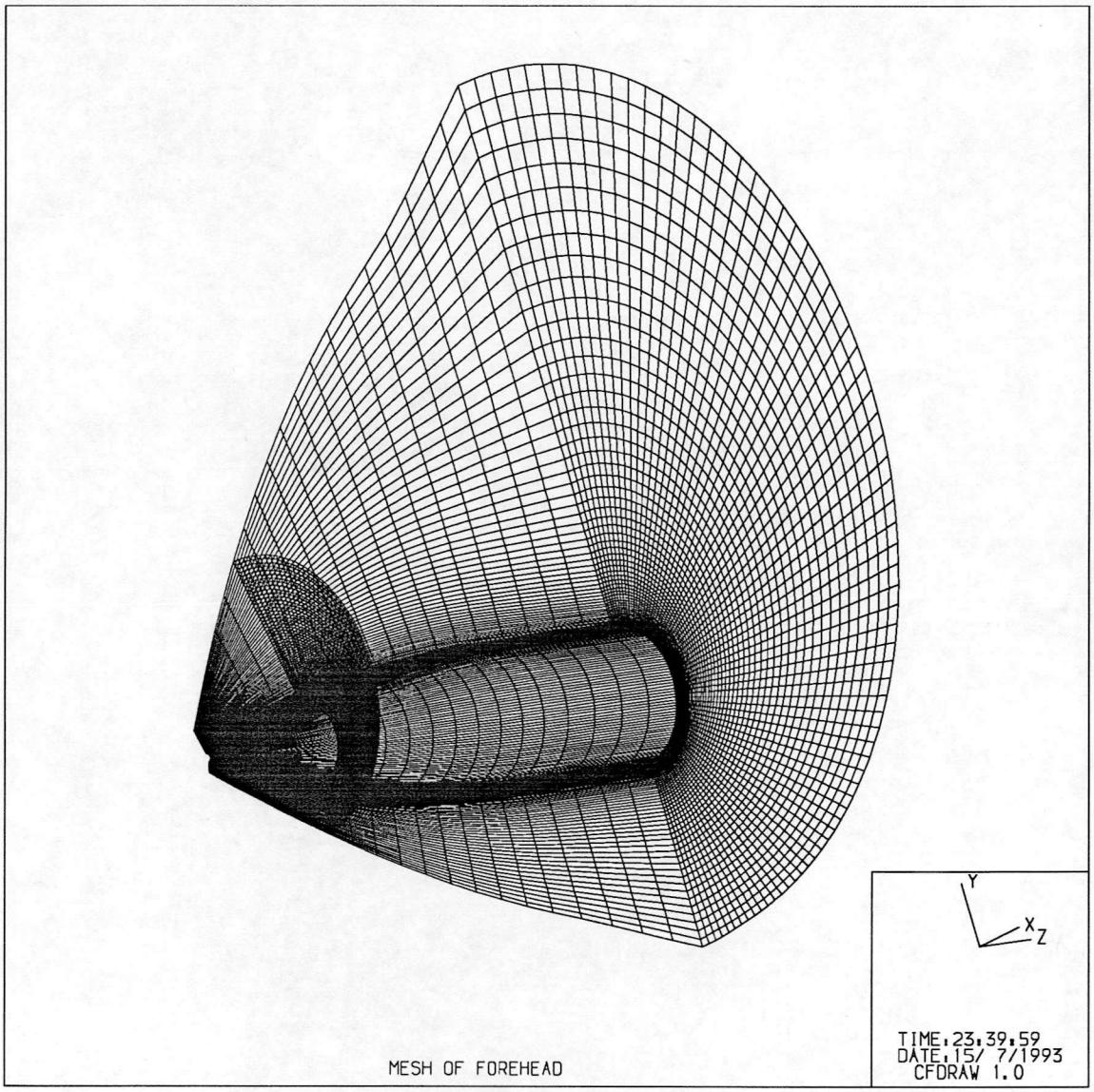


Fig. 6.2.1 Detailed mesh of the fore body, first twenty I-stations.

This test case was chosen because it had been considered to be particularly demanding. From Fig.6.2.1 it can be seen that the area around the surface is much denser than rest of the computed domain since it is in the boundary layer that we are most interested.

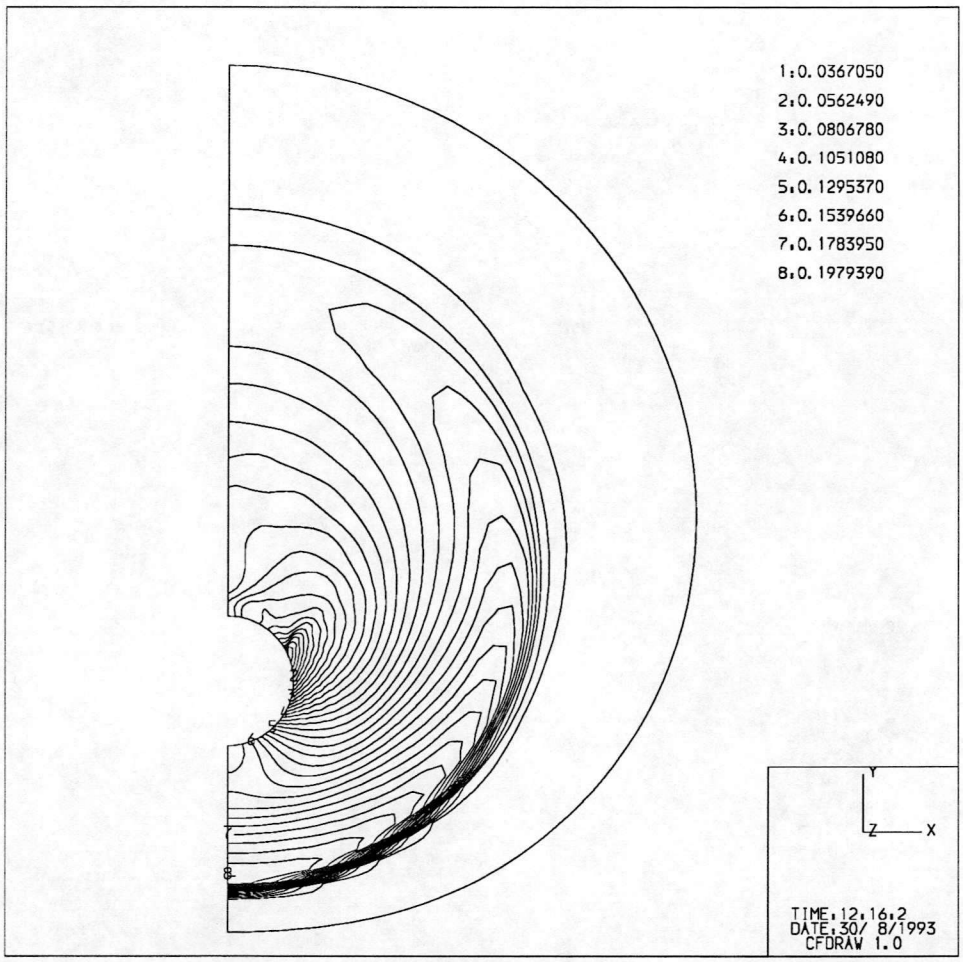
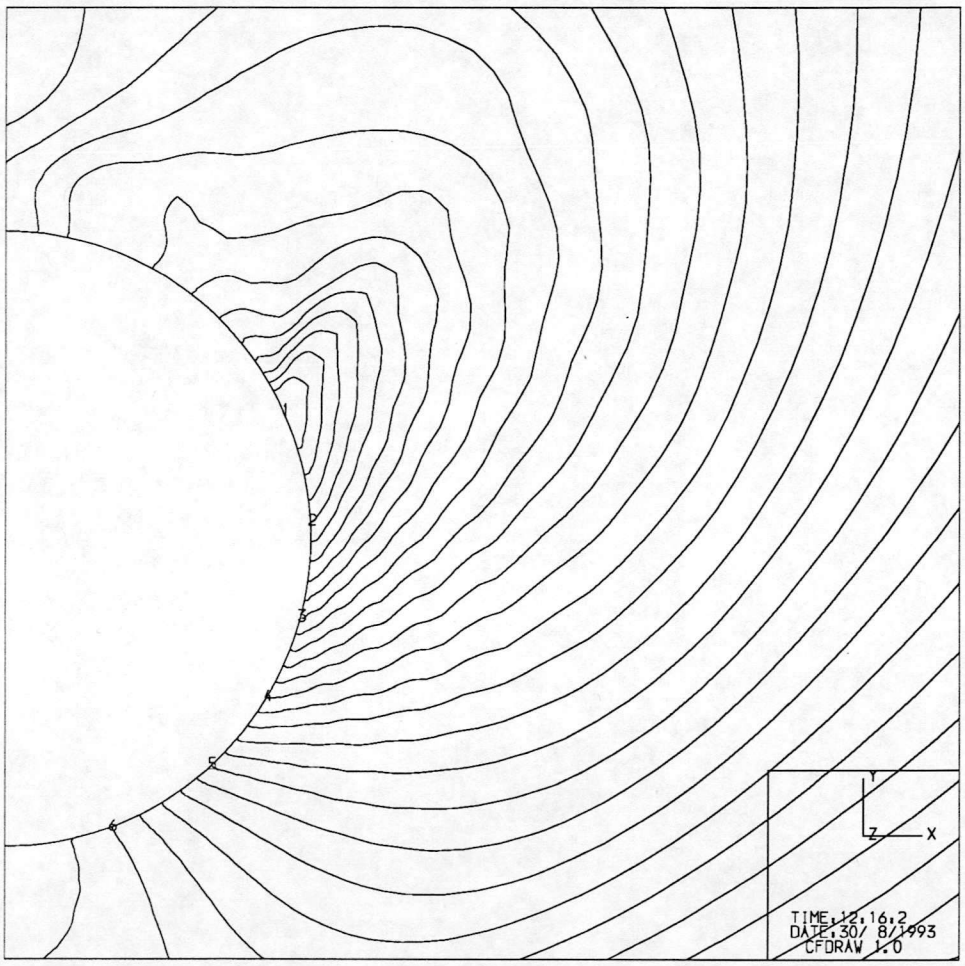


Fig. 6.2.2 Pressure contours at I=20 (above), and enlargement around the wall (below)



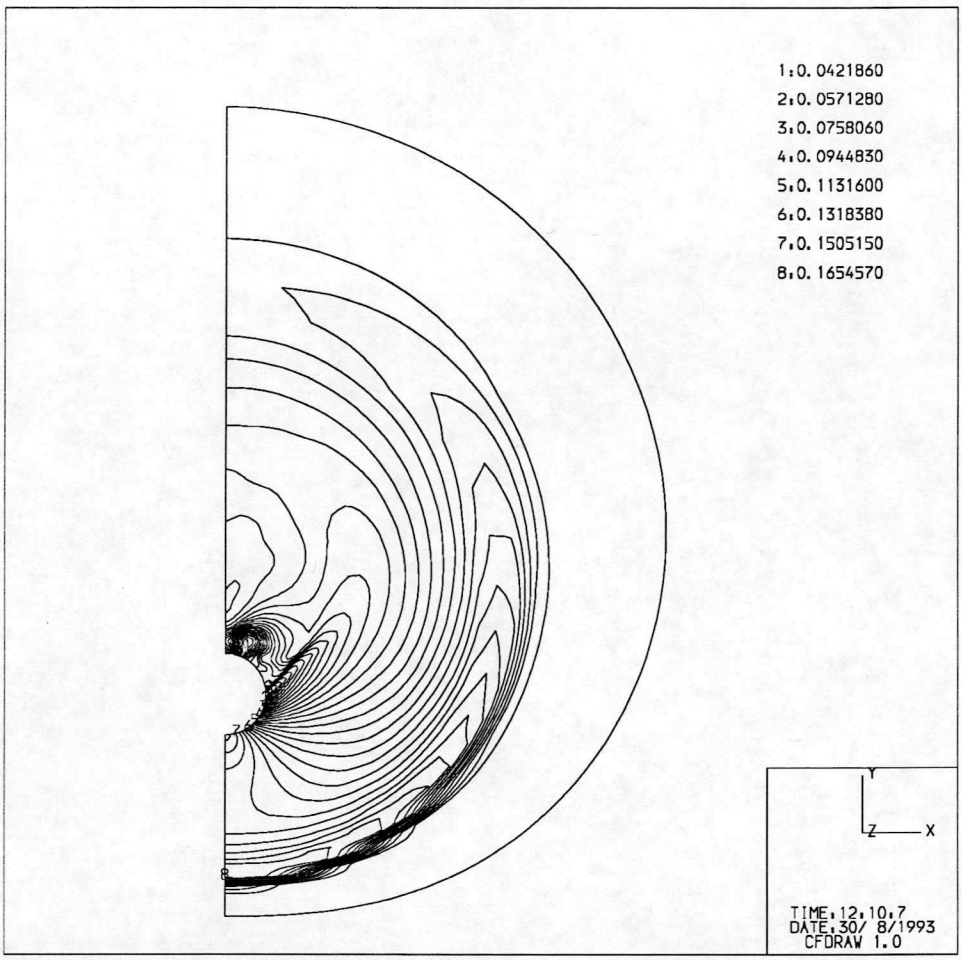
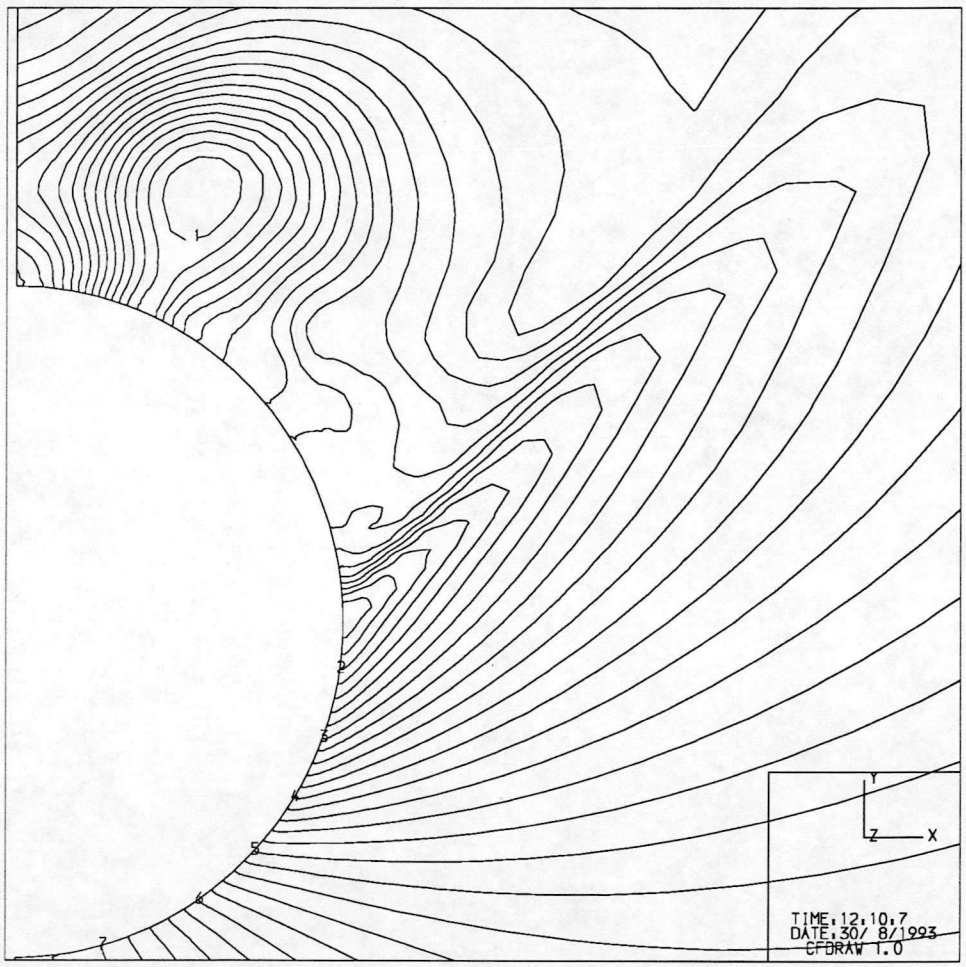


Fig. 6.2.3 Pressure contours at I=30 (above), and enlargement around the wall (below)



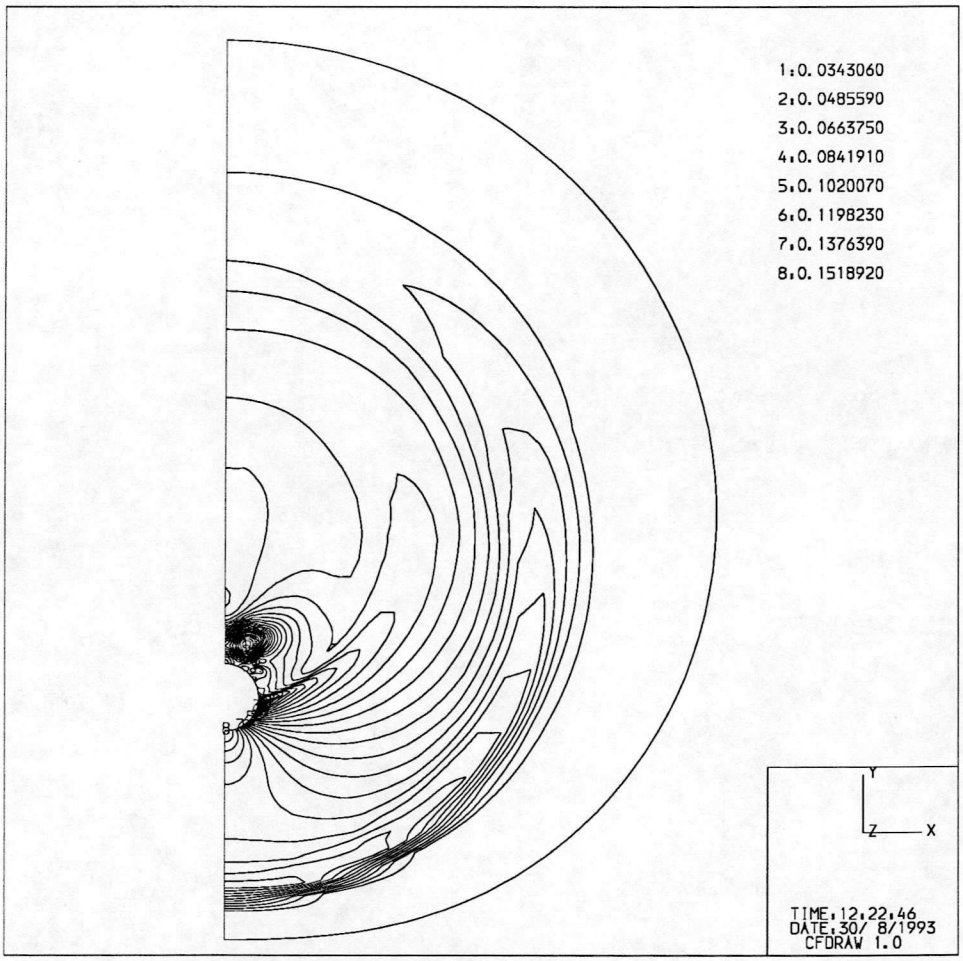
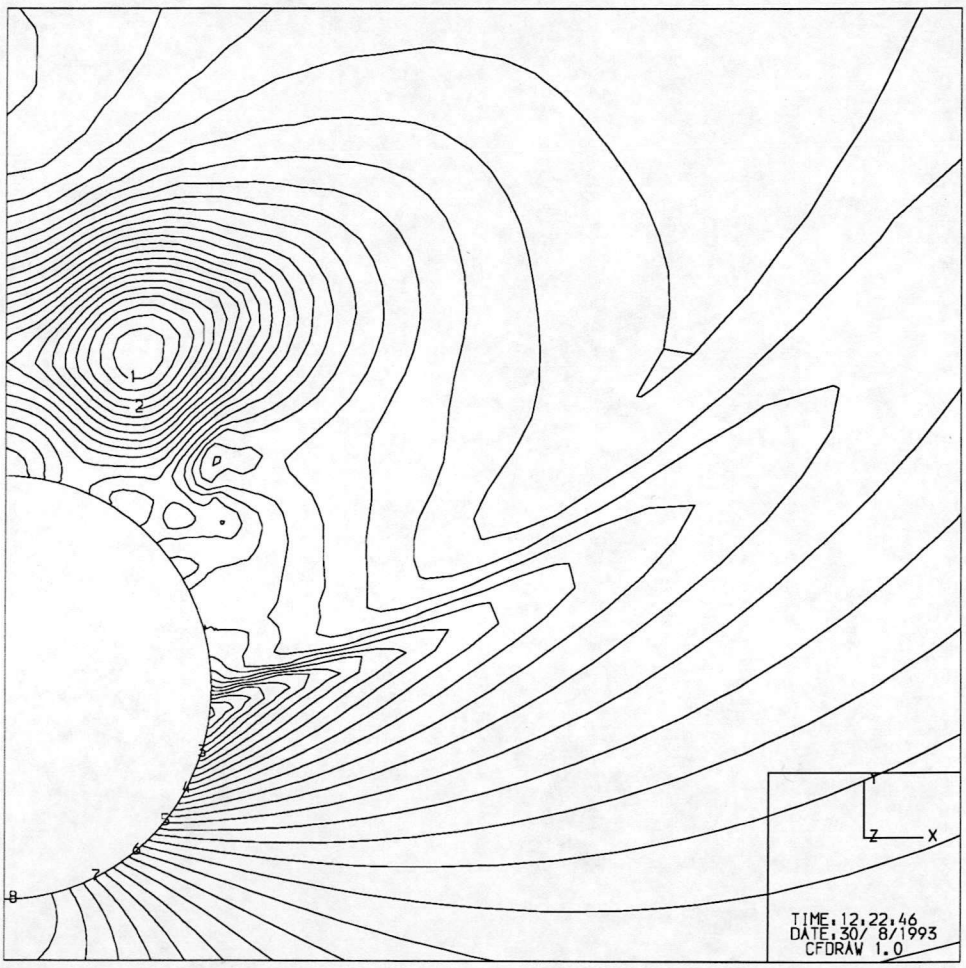


Fig. 6.2.4 Pressure contours at I=40 (above), and enlargement around the wall (below)



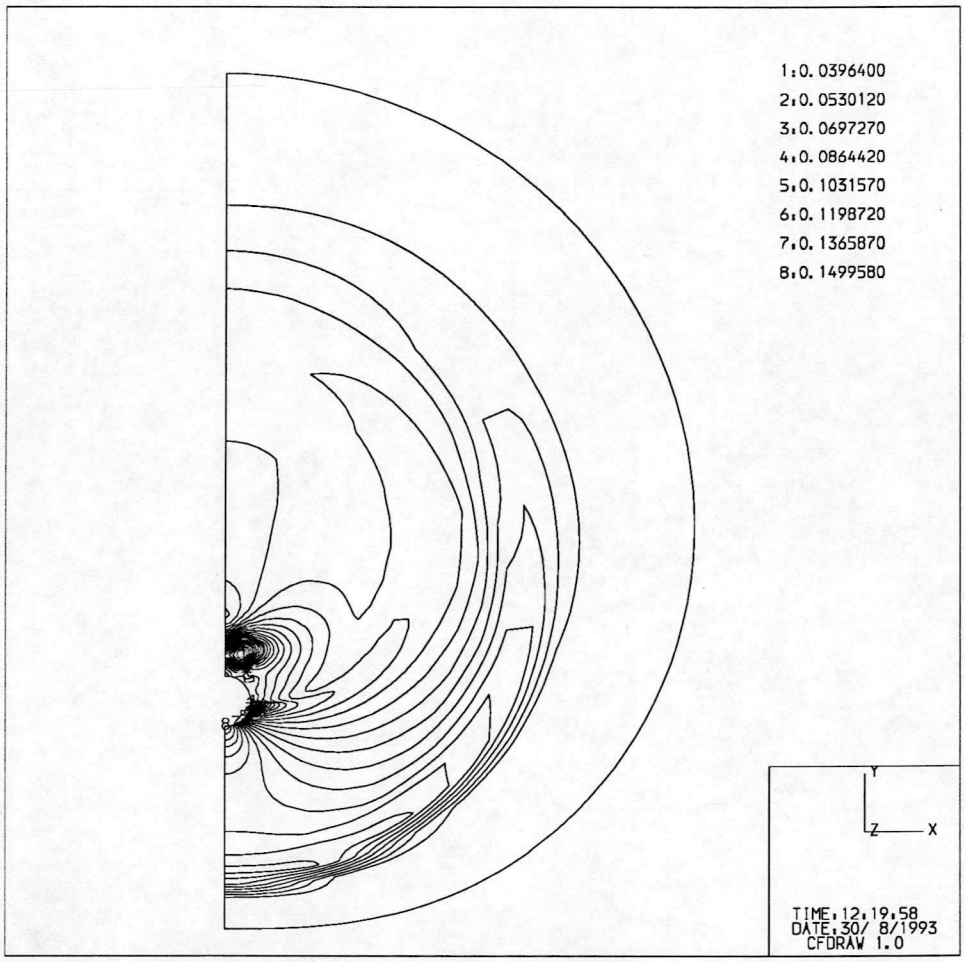
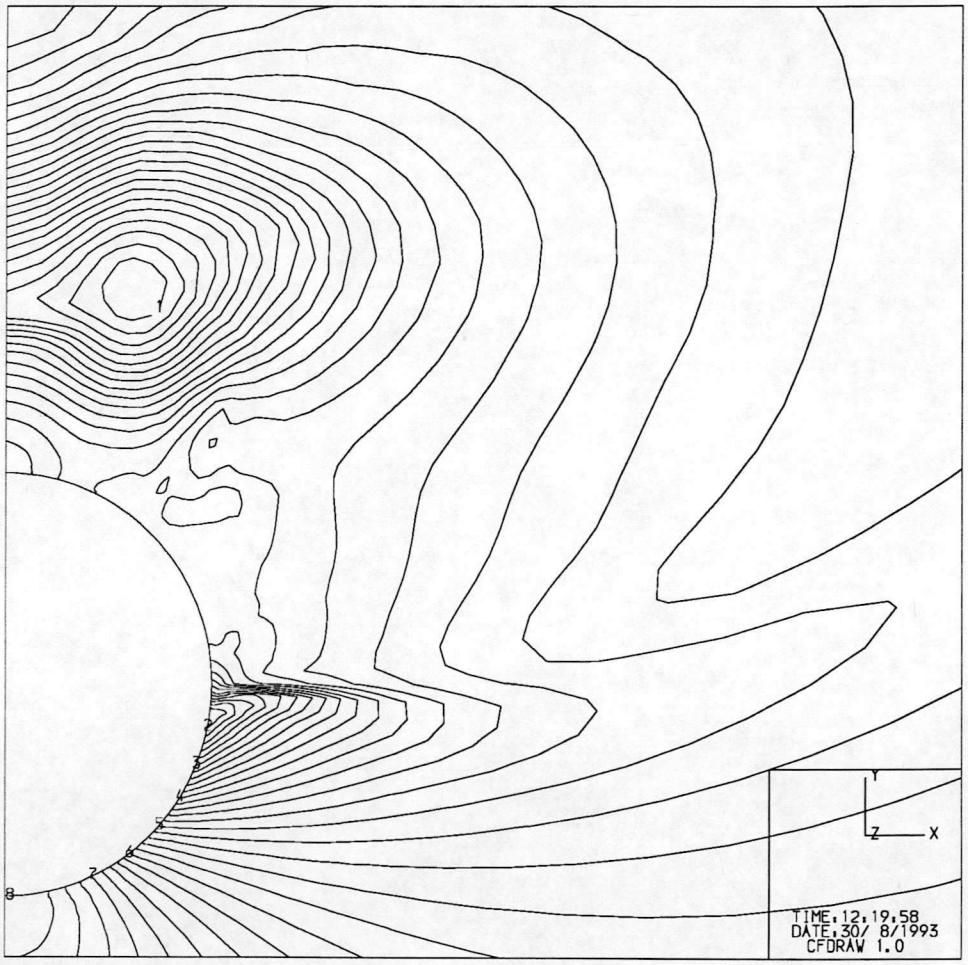
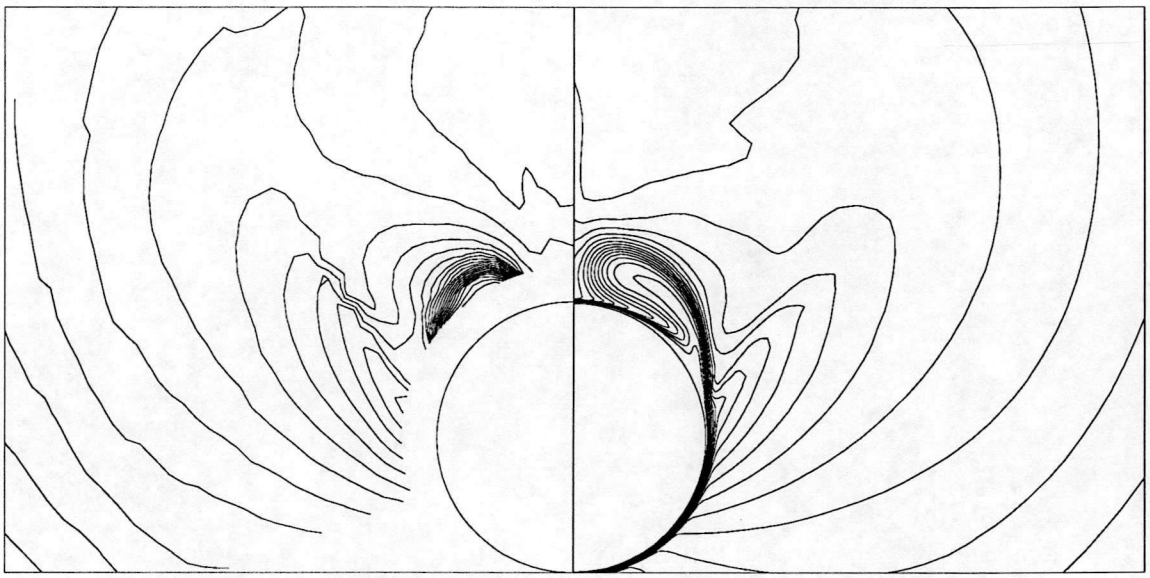


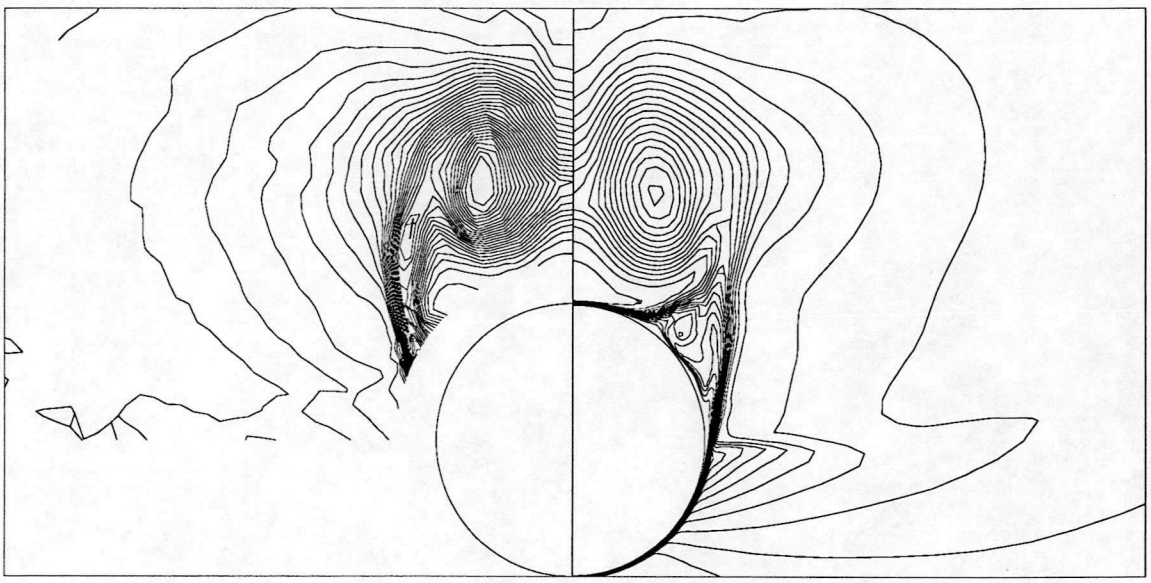
Fig. 6.2.5 Pressure contours at I=50 (above), and enlargement around the wall (below)





Experimental measurement Turbulent NS3D

Fig. 6.2.6 The Pitot pressure results: comparison between experiment and computation. M=2.5, AoA=14°, Re=4x10^6/ft, x/D=5.5



Experimental measurement Turbulent NS3D

Fig. 6.2.7 The Pitot pressure results: comparison between experiment and computation. M=2.5, AoA=14°, Re=4x10^6/ft, x/D=11.5

Figures 6.2.2-6.2.5 are the pressure contours at I = 20, 30, 40 and 50, respectively. It can be seen that the pressure distribution in the crossflow develops and the details of it around the surface, the formations of the vortex and the separation in particular. Figures 6.2.6 and 6.2.7 show the comparison between the computed pitot pressure and the experimental measured pitot pressure at two axial locations (x/D=5.5 and x/D=11.5). The similar plots can be found in reference [9] in which the comparison is between TPNS3D's results and the same experimental measurements.

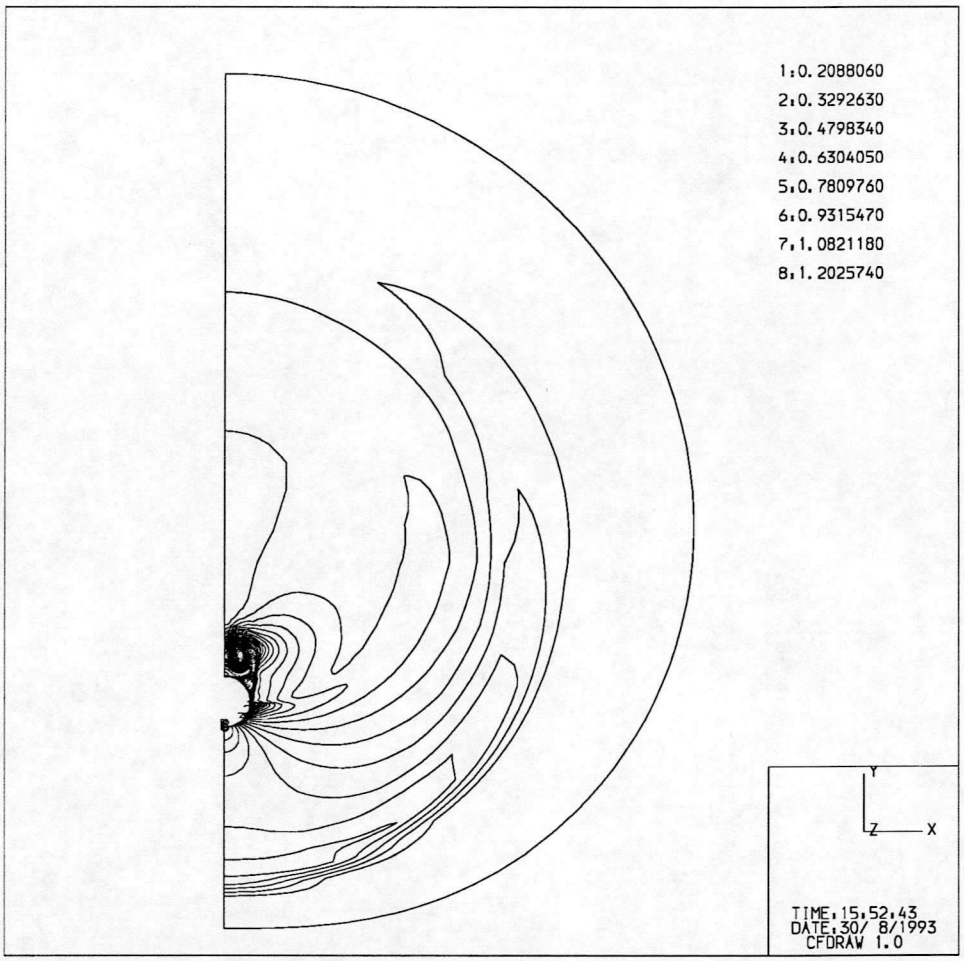
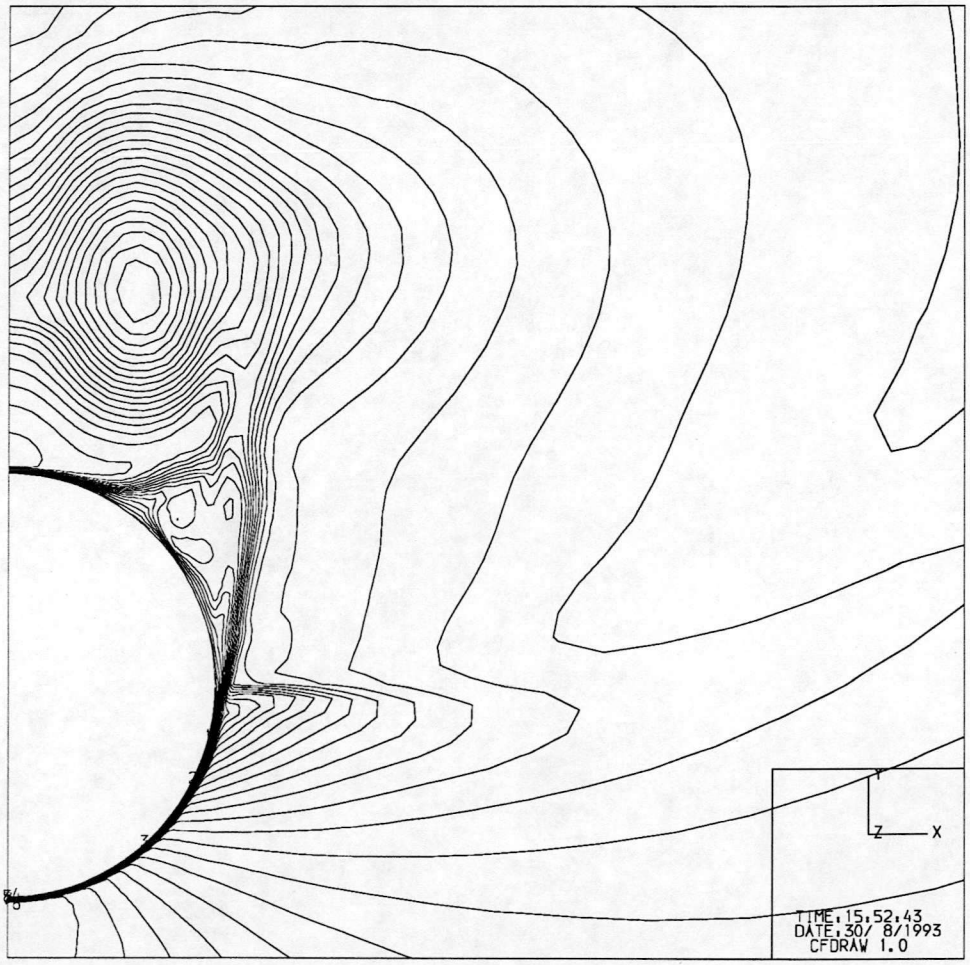


Fig. 6.2.8 Density contours at I=50 (above), and enlargement around the wall (below)



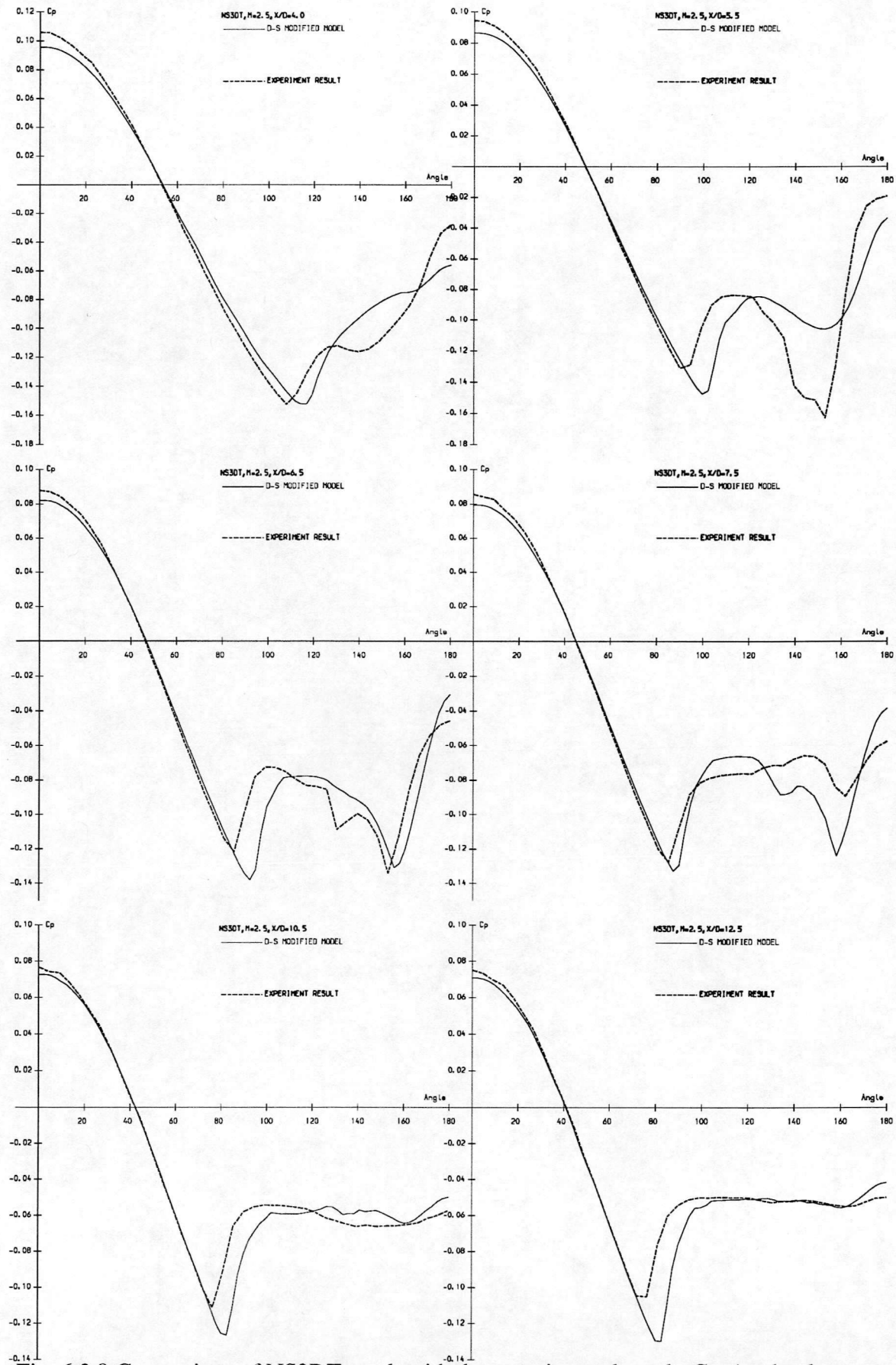


Fig. 6.2.9 Comparison of NS3DT result with the experimental result: Cp-Angle plots.

Fig. 6.2.10 Temperature contours at I=50 around the wall.

Figure 6.2.8 shows the density contours at I = 50 which looks very similar to the pitot contours shown in figure 6.2.7. The enlargement of the temperature contours around the surface is shown in figure 6.2.10. Cp-angle plots of the circumferential surface pressure at six axial locations are shown in figures 6.2.9, from which it can be seen that a close agreement between the NS3DT result and experimental measurement is reached. Reader may be reminded to those in TPNS3D's comparisons of the same test case, by which it can be found that the fully three dimensional calculation does give better result in the turbulence modelling.

7.CONCLUDING REMARKS

A turbulent NS3D program has been successfully completed. Two test cases have been used to validate the program. The results of NS3DT from Baldwin-Lomax model and Degani-Schiff modification of the model are in good agreement with the experimental measurements, the Johnson-King model is still need to be validated by a suitable test case.

The program itself has been modified to provide more efficiency and easier understanding, with the storage required being reduced. The major improvement in terms of numerical technique is to use temperature instead of pressure for the calculation. This can avoid the non-physical phenomena such as occurrence of negative pressure, and give better stability for the numerical technique. Pressure always gives stronger discontinuity in

a numerical method than temperature does, which more easily causes oscillations in the vicinity of a shock wave, and is sensitive to small disturbances in a pressure loss area where a negative pressure could be easily produced. Some higher order schemes use the character of pressure for dissipation which may cause other non-ideal solution in capturing a shock wave. Therefore, from the author's point of view, using temperature in this scheme seems sensible.

The original Baldwin-Lomax model was proposed about fifteen years ago, though it is a simple turbulence model, it can still give good predictions. One important point of applying this model successfully to some practical problems is to give a fine enough grid in the boundary layer, both in radical and circumferential directions, so that it can find the right calculated values for the model, such as ω, and Γ_{max}, Y_{max} in particular, because the peak value of $\Gamma(y)$ can be easily missed out.

The Degani-Schiff's modification of the B-L model is specially designed for some cross flow separation cases, such as case II in which there is a transition section from laminar flow to turbulent flow, and the vortex is formed and moving away from the wall in this transition section. The function $\Gamma(y)$ then has two or more peaks in the leeward side after the primary separation point. However it has been found that the introduction of the value Y_{cutoff} is not suitable for case II due to relaminarization after separation. The test case was actually computed with the modified model without this Y_{cutoff} value. The treatment of the area in which the two peaks merge should be studied further since in the area for this case the eddy viscosity does need to be larger for simulating the separation which causes low pressure and subsequent shock compression.

The Johnson-King model is still to be tested. The model is designed for upstream separation, and a ordinary differential equation for the maximum Reynolds shear stress needs to be solved. For a fully three dimensional extension of the model, it should be noted that the maximum Reynolds shear stress is not a field variable, instead, it is only the maximum value of the Reynolds shear stress along each ray from the object's surface. It is therefore not clear that the partial differential equation of three dimension can be used for the solution of the maximum value of the Reynolds shear stress. The model has been coded in the program by adopting the steps mentioned in references [6,7], and a suitable test case must be found for validating the turbulence model.

For fully three dimensional flows, the computation is very time-consuming, which is always the major concern. Hereafter we list the CPU time used for one iteration of grid size 55x65x75 of this code:

Computer System	NS3DT (Turbulence)	NS3DT(Laminar)
IBM RS/6000 320H	114	106
IBM RS/6000 550	69	63
SUN SPARC 10/20	105(Swapping)	81(Swapping)
Cray YMP	32.8	30
8 Processors in i860	55	50

Table 7.1 CPU time of various systems in seconds.

For the first test case some 8,000 iterations were used by the code on iPSC/860, and the second test case more than 10,000 iterations for reducing the residuals' summations down by four orders on Cray YMP. Whereas by using TPNS3D only about a third of the CPU time is needed for the same size of grid to better convergence.

The size of the code for the grid is 27Mbytes for single precision and 7Mwords for double precision in Cray YMP.

Acknowledgements: This work has been funded by SERC / MoD grant GR/H90035.

REFERENCES

[1].Qin, N., "User's Guide for the NS3D Code", Report of Department of Aerospace Engineering, University of Glasgow, April, 1991

[2].Baldwin, B. and Lomax, H., "Thin-Layer Approximation and Algebraic Model for Separated Turbulent Flows", AIAA Paper 78-257, Jan. 1978

[3].Cebeci, T. and Smith, A.M.O., "Analysis for Turbulent Boundary Layers", Academic Press, New York, 1973

[4].Degani, D. and Schiff, B., "Computation of Turbulent Supersonic Flows around Pointed Bodies Having Crossflow Separation", J. Comp. Phys., Vol. 66, No. 1, 1986, pp.173-196

[5].Johnson, D. A. and King, L. S., "A Mathematically Simple Turbulence Closure Model for Attached and Separated Turbulent Boundary Layers", AIAA Journal, Vol. 23 No. 11, Nov. 1985, pp.1684-1692

[6].Abid, R., Vatsa, N., Johnson, D. A. and Wedan, B. W., "Prediction of Separated Transonic Wing Flows with Nonequilibrium Algebraic Turbulence Model", AIAA Journal, Vol. 28, No., 1988, pp.1426-1439

[7].Gee, K., Cummings, R. M. and Schiff, L. B., "Turbulence Model Effects on Separated Flow About a Prolate Spheroid", AIAA Journal, Vol. 30, No. 3, March 1992, pp.655-664

[8].Spekreijse, S.P., "Multigrid Solution of the Steady Euler Equations", CWI Tract, Centre for Mathematics and Computer Science, Netherlands, 1988

[9].Van Leer, B. "Upwind-Difference Methods for Aerospace Problems Governed by the Euler Equations", Lecture Note in Applied Mathematics, Vol. 22, Part II, pp.327-336

[10].Anderson, W. K., Thomas, J. L. and Van Leer, B., "A Comparison of Finite Volume Flux Vector Splittings for the Euler Equations", AIAA Journal, Vol. 24, No. 9, pp.1453-1460

[11].Jin, X., "A Three Dimensional Parabolized Navier-Stokes Solver with Turbulence Modelling", G.U. AERO Report 9315, July, 1993

[12].Jin, X., "CFDRAW — A Pre- and Post-processor for Three Dimensional Fluid Dynamics", User's Guide for Department of Aerospace Engineering, University of Glasgow, Sept., 1993